Various facilities are often provided in programming systems (such as header cards to change names in some assemblers); but these are too limited to allow us to take the appropriately relaxed attitude that there need be no official version of a system. It is not very far fetched to consider that languages should have sufficient syntax and basic processes to permit their easy self manipulation. Many partial solutions toward this already exist.

This paper is no place for a technical discussion of the various ways languages should evolve to aid the problems of documentation and banish the need for officialdom (nor is the author prepared, to be honest). Yet, it would be a mistake to accept as given many of the constraints that currently make documentation a headache to prepare and to use.

JOVIAL AND ITS DOCUMENTATION

Christopher J. Shaw, System Development Corp., Santa Monica, Calif.

A Quick Look at JOVIAL

Jovial is a general-purpose, procedure-oriented, and largely computer-independent programming language. It is intended primarily for professional programers, since it was designed and implemented by SDC to produce programs for large, computer-based, military command and control systems. It is currently being used for this purpose in the development of several such systems.

Jovial is derived from Algol 58, with the following major extensions: an input-output notation; a more elaborate data description capability; the ability to manipulate fixed-point numeric values; and the ability to manipulate symbolic and other non-numeric values, including machine-language symbol-segments.

The design of Jovial was mainly dictated by three requirements. The language must: (1) be able to express the wide variety of procedures likely to be needed in command and control problems; (2) be economical of programming effort; and (3) be as machine-independent as possible, to be suitable as a corporate standard. Additional requirements, derived from experience with command and control programming, were: compilers should accept system environment information—data descriptions and storage allocation parameters—from a Compool (*com*munication *pool*); imperative sentences should avoid expressing details that are often changed—e.g., those relating to data organization, coding and scaling—to minimize the program modification chore; compilers should produce efficient code, to meet the space/time constraints of real-time systems.

Jovial development has been supported partly by corporate and partly by contract funds. Currently, Jovial compilers are operating on and for: the IBM 7090; the IBM AN/FSQ-31v and the closely related AN/FSQ-32; the IBM AN/FSQ-7; the Philco 2000; and the CDC 1604. The 7090, 2000, and 1604 compilers have been made available through the computer users groups: SHARE, TUG, and CO-OP.

Jovial compilers are written and maintained almost entirely in Jovial. They consist of two main parts: first, a Generator, which transforms Jovial programs into an Uncol-like Intermediate Language; second, a Translator, which further transforms them into machine language. Translators ordinarily incorporate a complete, symbolic assembly phase. Compilers for the Q-7, the 2000, and the 1604 use essentially the same, common Generator and Intermediate Language. Thus, instead of producing three, complete compilers, it was only necessary to produce one Generator and three Translators.

Jovial compilers range in size from fifty to sixty thousand machine instructions, and it has been estimated that about five man-years of work are needed to write a new translator and get a compiler running on a new machine.

SDC began work on Jovial in February 1959, in Paramus, N. J. A year later, in February 1960, a Jovial Interpreter (consisting of a Generator and an Intermediate Language interpreter) was running on the IBM 709; by October 1960, a compiler for the IBM AN/FSQ-31v was running on the 709;[1] and by December 1960, a compiler for the 709 was running on the 709.

Even before these compilers were fully operational, work had begun on improvements—to the compilers, and the language as well. The current compiler for and on the IBM 709, 7090 was operational in March 1961; the compiler for and on the IBM AN/FSQ-31v was operational in October 1961.

Meanwhile, in March 1960, work began in Santa Monica on another Jovial compiler, for the IBM AN/FSQ-32. But with the cancellation of the contract involving this computer, in July 1960, the project was switched over to the IBM AN/FSQ-7 as the target machine.

At about that time, SDC decided to adopt Jovial as a corporate-wide, standard programming language. This decision involved redesigning the language to make it more nearly computer-independent. SDC also decided to produce a compiler-independent, common Generator, for use with the Q-7 and subsequent compilers. (Unfortunately, due to schedule pressures, the 7090 and Q-31v compiler projects could not await the completion of this common Generator, and cross-continent attempts to eliminate source language differences were not entirely successful.)

Prior to the completion of the compiler for the Q-7, compilers for the Philco 2000 and the CDC 1604 were begun—in October 1960 and in December 1960—in Santa Monica. The Q-7 compiler was operational in June 1961; the 2000 compiler was operational in October 1961; the 1604 compiler was operational in January 1962.

More recently, the Q-31v compiler developed in Paramus has been adapted, in Santa Monica, for use with the closely related IBM AN/FSQ-32 computer, used by SDC's Command Systems Laboratory. In addition, the Translator portion of this adapted compiler is currently being rewritten, to make it compatible with the common Generator. Also in Santa Monica, an experimental compiler—accepting a subset of Jovial—is being written; initially on and for the IBM 7090 but ultimately, perhaps, for smaller machines.

In SDC's Washington Division, the compiler staff there is modifying the 1604 compiler for the CDC 1604A, a slightly

[1] It produced Q-31v programs that were, in the absence of a Q-31v, interpretively executed on the 709.

different model. And in SDC's Lexington, Massachusetts office, the 2000 compiler is being fitted into the Philco SYS operating system.

In addition to this SDC activity, Computer Associates, Incorporated, of Woburn, Massachusetts, is developing a syntax-directed compiler for the Burroughs D-825 computer, which will process JOVIAL.

Despite all this far-flung activity, the agency within SDC primarily responsible for JOVIAL maintenance is the JOVIAL Compiler Staff in Santa Monica—about 16 people, headed by Gene Gordon. The Compiler Staff, which is part of SDC's Information Processing Directorate, under Don Madden, is responsible for maintaining the following items (and the official documentation relating to them): the language itself; the common Generator for the Q-7, 2000, and 1604 compilers; the 7090, 2000, and 1604 compilers. These arrangements may seem complex, but the philosophy behind them is fairly simple: items with more than one (actual or potential) user are maintained by the JOVIAL Compiler Staff; items with just one user are maintained by the using agency.

JOVIAL Documentation

My collection of documents on JOVIAL weighs almost 50 pounds and stands almost two feet high. Properly sorted into file boxes, it occupies an entire shelf in my bookcase.

While the collection is fairly comprehensive, it is by no means complete. It contains no compiler-listings (these are not published as documents), and many obsolete, superseded, or unimportant documents are missing. Nevertheless, this collection of documents represents a major result of the almost four years of effort spent by SDC in developing JOVIAL. (The other major, tangible result is, of course, the various JOVIAL compilers themselves.)

This mountain of literature was produced by a multitude of authors from many different SDC internal organizations, each with slightly different needs and criteria for documentation. Consequently, individual documents vary widely in quality, and the collection as a whole is not a particularly well-integrated body of literature. To survey it adequately would be a mammoth task. It would also be unnecessary. Half the collection consists of informal notes and memoranda meant for temporary, internal use only. These need not be considered. The remaining documents, some of which are listed in the bibliography, have been reviewed for technical content and are generally available for outside distribution.

These documents can, in turn, be divided into two classes: official; and unofficial. Documents are official whose technical accuracy *and* maintenance are the responsibility of the agency responsible for implementing or maintaining the items treated by the document. Those that do not fit this category are unofficial. Unfortunately, this useful distinction is not always made explicit, so it is sometimes necessary to guess which is which.

Of the official documents on JOVIAL, perhaps the two most interesting and controversial, in terms of programming language documentation in general, are *The JOVIAL Grammar and Lexicon* [4], which is a formal description of the language, with syntactic definitions, and *The JOVIAL Primer* [6], which explains the use of the language. As the author of these documents, I can safely discuss them. The principal, practical fault of these two documents is that they both describe the "official" version of JOVIAL, which none of the compilers have yet completely implemented.[2] Consequently, it was necessary to document the JOVIAL dialect that each compiler accepts, even though the differences are minor in most cases. This need has resulted in five additional language description documents, and the programmer must study the appropriate one in order to successfully write and compile JOVIAL programs on a specific machine. This situation seems to plague most programming languages where there is an "official" version and several implemented dialects. It could probably be avoided by careful segmenting of the "official" description so that inappropriate segments can easily be removed and changes or supplements inserted to form the description of some implemented version of the language.

For quite a while, the only available document describing JOVIAL for the Q-7, 2000, or 1604 programmer was *The JOVIAL Grammer and Lexicon*, [4]. This document is a formal, syntactic description intended mainly for implementers, who usually appreciate its rigor—after they learn the syntax metalanguage. But it was never very popular as a programmer's manual. Programmers seem to prefer a more informal treatment.

As well as being easier to understand, an informal description is usually easier to write, maintain and modify than a formal, syntactic description, since it requires less sophistication on the part of the person or persons involved. It would seem, then, that formal, syntactic descriptions of programming languages are only needed where the language designers are not the language implementers. (This usually happens, though, when a language is implemented for more than one machine.) In any event, such descriptions should be kept out of the hands of programmers, if at all possible.

Even after the publication of *The JOVIAL Primer* [6] had supplied the experienced programmers with a useful description of the language, two important classes of potential users were neglected. There were no documents tailored specifically to the needs of either the programmer-trainee, who usually is not ready for all the subtleties of using the language, or the part-time non-professional programmer—the dilettante—who does not ordinarily need all the capabilities the language affords. At this writing, however, two documents, [8] and [9], fitting the needs of these two groups are in the final stages of preparation. They should be available by the time this paper is published.

A Selected Bibliography on JOVIAL

The documents listed here were chosen from a larger list of 75 which in turn were chosen from the many that have been published on JOVIAL because of their current general interest. Each listing contains a brief annotation, giving the status of the document (e.g., official, unofficial, interim), its type (e.g., reference manual, primer, report), its intended audience (e.g., the interested public, the programmer, the implementer, the maintainer), its topic (e.g., the language, the compiler, etc.) and whether or not it contains syntax definitions, program listings, or flow charts.

All of the documents listed here were published by the System Development Corporation.

SDC's document numbering scheme is quite complex; but it is meant to encourage the publication of related documents under the same basic number. Document numbers have the following format: series-location-number/volume/revision. Thus, FN/LX-393/212/02 reads: FN-series, Lexington, number 393, volume 212, revision 2. The series codes are not of particular interest here. The location codes are, however, since documents are best ordered from the location where they are published. No location code—2500 Colorado Ave., Santa Monica, Calif.;

[2] Although all of it has been implemented in one compiler or another.

LX—45 Hartwell Ave., Lexington, Mass.; LO—567 Winters Ave., Paramus, N. J.; WD—5821 Columbia Pike, Falls Church, Va.

1. Wilkerson, M. 7090 JOVIAL compiler and checker user's manual. FN-LO-501, Apr. 1961, 39 pp. An official reference manual for the programmer on the 7090 compiler and the Checker, a source language debugging tool for use with the compiler.
2. Spierer, M. The 7090-JOVIAL-to-7090 compiler system. FN-LO-503, July 1961, 173pp. An official reference manual (with flow charts) for the maintainer describing the 7090 compiler.
3. Howell, H. L., Isbitz, H., and Schwartz, J. I. Documentation of the JOVIAL language and compiler for the IBM 7090 computer; The JOVIAL language for the 7090 computer; Techniques of input-output for JOVIAL on the 7090 computer; Available subroutines for JOVIAL on the 7090 computer; Operation of the JOVIAL compiler on the 7090 computer; Error messages of the 7090 JOVIAL compiler; Reporting and responding to problems with the 7090 JOVIAL compiler; The communication pool for the 7090 JOVIAL compiler. FN-6223 & supplements; about 150 pp., Jan. 1962. An official reference manual, for the programmer, on the 7090 compiler and the language it accepts.
4. Shaw, C. J. The JOVIAL grammar and lexicon. TM-555/002/01 and TM-555/002/01A, June 1961, 77 pp. An official reference manual, for the implementer, specifying the language, with syntactic definitions.
5. Shaw, C. J. A programmer's introduction to basic JOVIAL. TM-629, Aug. 1961, 39 pp. An official reference manual for the experienced programmer, briefly describing, with syntactic definitions, the basic elements of the language.
6. Shaw, C. J. The JOVIAL primer. TM-555/003/00, Dec. 1961, 216 pp. An official primer, for the programmer, on the use of the language with syntactic definitions, examples and exercises.
7. Cartmell, D. J. The Intermediate Language (IL) table. TM-555/050/00, Jan. 1962, 15 pp. An official reference manual for the implementer and maintainer on the Intermediate Language, in which JOVIAL imperative statements are encoded between the Generator and Translator portions of the Q-7, 2000, and 1604 compilers.
8. Perstein, M. H. JOVIAL for the dilettante, Part 1. TM-555/061/00, Oct. 1962, 40 pp. An official primer and reference manual for the nonprofessional programmer describing a limited subset of Jovial and some standard input-output procedures.
9. Kennedy, P. A simplified approach to JOVIAL (a training document). TM 780/000/00, Sept. 1962. 387 pp. An interim primer and reference manual for the beginning programmer on the language, its use and the restrictions on its use imposed by the various compilers.
10. Perstein, M. H., Clark, E., and Hayes, E. Implementation of JOVIAL in SSRL. TM-555/200/00 and TM-555/200/00A, Dec. 1961 and Mar. 1962, 37 pp. An official reference manual for the programmer and the implementer/maintainer describing the language accepted by the 2000 compiler in terms of its differences from that described in [4].
11. Clark, E. Phase 1 of the Philco 2000 JOVIAL translator. TM-555/211/00, Aug. 1962, 60 pp. An official reference manual for the maintainer describing the first phase of the Translator part of the 2000 compiler.
12. Phase 2 of the Philco 2000 JOVIAL translator. TM-555/212/00, Aug. 1962, 326 pp. An official reference manual for the maintainer on the second (and final) phase of the Translator part of the 2000 compiler.
13. 1604 JOVIAL compiler, program description of the translator pass 1. TM-555/302/00, June 1962, 74 pp. An official reference manual for the maintainer, describing the first pass of the Translator portion of the 1604 compiler.
14. Jacoby, L. 1604 JOVIAL compiler, JOVIAL programming guide. TM-WD-555/301/00, Oct. 1962, 90 pp. An official reference manual for the programmer on the language accepted by the 1604 compiler.

NELIAC

M. H. Halstead*, U.S. Navy Electronics Laboratory, San Diego, California

The following is an account of current documentation on the NELIAC Language.

Administrative Aspects of the Language

Sponsor. The Neliac language has been sponsored primarily by the Navy Electronics Laboratory, with valuable assistance from the Navy Postgraduate School at Monterey, the University of California at Berkeley and the Signal Corps at Fort Huachuca. In each case, however, it has been more a matter of the enthusiasm of individual supporters at these installations than of official sponsorship. Perhaps this factor has contributed to an increased flexibility of design which has proved advantageous.

Key Dates. Work on the first Neliac compiler was started in July, 1958, and completed within the following six months. Specifications for the language were determined concurrently.

Maintenance. Maintenance of Neliac compilers is officially nonexistent. In the view of some nonusers this is a fatal deficiency, while in the view of some users it is an unqualified advantage. In order to understand the latter point of view, three of the special characteristics of Neliac compilers should be recognized. First, Neliac compilers are self-compilers. Consequently, they are all written in the Neliac Language. Second, most Neliac compilers have been kept relatively short and simple, with very few rules or exceptions to rules. Third, most Neliac compilers have become relatively fast, with compiling speeds of many thousands of computer words per minute. As a result of the first two factors, a programmer who is familiar with the *Language* can read, understand and improve any given *compiler.* As a result of the third factor, he can recompile an improved version of a compiler quite cheaply, since some of them actually recompile in less than one minute.

Machines for Which Implemented. Neliac compilers have been implemented for machines shown in Table 1. Most dates are approximate.

* The opinions and assertions contained herein are the private ones of the writer, and are not to be construed as official, or as reflecting the views of the Navy Department or the naval service at large.