# Taking M Beyond the Year 2000

*by Art Smith*

## Introduction

The following document is an abbreviated form of a Standing Document of Subcommittee 16 (Object Oriented Language) of the MUMPS Development Committee. This document states much of the purpose of this new subcommittee and gives an overview of the direction the subcommittee expects to take at this time. It is being presented here to keep the entire M community informed of the intentions of this new subcommittee and to invite comment and direction. If you wish to have your comments addressed formally by the subcommittee, please submit your comments in writing to MDC Subcommittee 16, 1738 Elton Road, Suite 205, Silver Spring, MD 20903. These comments will be entered into the MDC Document Registry and will be addressed by the subcommittee at a future MDC meeting.

In short, this document suggests the creation of a new object oriented programming language described by a new language Standard (ANSI/MDC X11.7). This new language would be interoperable with MUMPS (ANSI/MDC X11.1), and is intended to be complementary with MUMPS, not a successor to it. A number of guidelines for the new language are included, based on the results of "straw poll" votes taken in a working group of the MDC. These votes are not binding, but are an attempt to reach consensus within the group. Subcommittee 16, Object Oriented Language, was formed at the March 19, 1997 MDC meeting to develop this language.

The original form of this document was a technical proposal submitted by the MUMPS Development Coordinating Committee for Europe (MDCC-E). This has been revised into the current form, which was approved as a Standing Document at the March 1997 MDC meeting. The full document (MDC registry number X11/SC16/97-2) includes an Appendix which details one approach to reaching the goals shown below. The Appendix has been omitted here for space, and because it represents just one approach being considered at this time, but the complete document can be requested by writing to the MDC Secretariat at the address given above.

Terms that are underlined in this document are formal metalanguage elements from the existing M standard (X11.1-1995). In particular, *eol* is end-of-line, *extsyntax* is the syntax for embedding other languages (such as SQL) in M, *expr* is an M expression, *svn* is an intrinsic special variable, and *ssvn* is a structured system variable. The acronym *YACC* is for "Yet Another Compiler Compiler" which is a compiler generation tool available in most UNIX systems. It takes a formal grammar specification (typically generated by a lexical analysis tool like LEX, and produces C code to compile this grammar).

It is important to note that this document is a Standing Document, which means that it has been endorsed by the Subcommittee as a whole (but not by the full MDC). It is not a technical proposal, however, and the language presented here is not expected to appear in any Standard specification approved by the MDC. Furthermore, this document represents the early stages of a work in progress and is subject to change without notice. The Task Group on the Object Model is in fact meeting by teleconference on a weekly basis at this time, and changes and developments are occurring more or less continuously. If you would like to participate in this Task Group, please contact Rick Marshall, Chairman of the Object Model Task Group. He can be reached by e-mail at toad@isc-sf.va.gov or by telephone at 206-764-2283.

The disclaimer below is formal notification of the impermanence of this document and is required by the MDC Constitution.

I hope you find this information helpful!

*Art Smith, MDC Chair*

* * *

## Disclaimer

"The reader is hereby notified that this document neither reflects MDC Standard specifications, nor any implied support by members of the MUMPS Development Committee or their sponsors. This document is endorsed by Subcommittee 16 of the MUMPS Development Committee. The Object Oriented Language Subcommittee intends this document to serve as a standing document indicative of their goals and directions and does not intend to advance it directly into any Standard specification. This document is dynamic in nature and may not correspond to the latest specification available.

"Because of the evolutionary nature of MDC Standard specifications, the reader is reminded that changes are likely to occur in the document release, herein, prior to a complete (re)publication of the MDC Standard." Copyright 1997 by the MUMPS

Development Committee. This document may be reproduced in any form so long as acknowledgement of the source is made.

"Anyone reproducing this document is requested to reproduce this introduction."

* * *

# MDC Document

**1. Identification of the Proposed Change**

**1.1 Title: Taking M beyond the Year 2000(V2)**

**1.2 MDC Proposor and Sponsor**

**Proposor:**
MDCC-Europe

**Sponsor:**
Frans S.C. Witte
DIC Information Consultants bv
P.O. Box 1572
/Bisonspoor 7005
3600 BN MAARSSEN, Netherlands
phone: +31-346-570 019
fax: +31-346-560 232
e-mail: FSCWitte@CompuServe.com

**1.3 Motion**
None

**1.4 History of MDC Actions**

| Date | Document | Action |
|---|---|---|
| 19 Mar 1997 | X11/SC16/97-2 | <this document> Approved as SC16 Standing Document, superseding X11/SC15/TG2/96-3 |
| Mar 1997 | X11/SC15/TG2/96-3 | Presented and approved by SC16 as standing document |
| Nov 1996 | X11/SC15/TG2/96-3 | Discussed in MDCC-E |
| 28 Sep 1996 | X11/SC15/96-11 | Voted on by SC15 |
| 26-28 Sep 1996 | X11/SC15/96-11 | Discussed in SC15/TG2/WG1 |
| 20-21 Jun 1996 | X11/SC15/96-11 | Discussed in MDCC-E (concentrating on issues in 7.3: mixing M1 and M2 glvns, some M "system objects") |
| Mar 1996 | X11/SC15/96-11 | Proposal as result of MDCC-E discussions. Presented in SC15/TG2. Not voted upon due to invocation of the time limit rule. |
| 11, 12 Jan 1996 | <several handouts> | MDCC-E meeting. Discussion of test group ideas. Input from membership |
| Nov 1995 | <no document> | Initial ideas expressed by MDCC-E membership. MDCC-E taskgroup appointed to present proposal to MDCC-E |

**1.5 Dependencies**
None known.

**2. Introduction**

**2.1 Needs**
It has been expressed numerous times that the M language, dating from the late sixties needs to incorporate a number of features that are considered state of the art (and that do occur in some or most other programming languages). Some topics have shown up during a couple of standardization cycles already, others have a more recent history. Examples of the former are "real IF-THEN-ELSE constructs" and "real block structuring." Examples of the latter are "object usage" and "object definition." The problem with many of these proposals is that they try to introduce concepts for which the language was never intended. This results in proposals that are often hard to understand, not obvious to implement, and almost impossible to explain to those who do not belong to the "in crowd," let alone newcomers.

On the other, hand the M language also contains a number of "features" that relate primarily to the original environmental circumstances, such as limited amount of memory and the interpretative nature of the language. Examples of such constructs are $STORAGE, $TEST, $NEXT, the naked-indicator, and end-of-line as scope-terminator. All this together does not contribute to the expansion of the market for M, which many consider of vital importance for M in order to survive as a language.

A number of proposals have tried to incorporate features that differ considerably conceptually from the current M concepts within the context of the M routine. A major problem with such

proposals is that the "block scope" concept of M relies heavily on the eol. This has resulted in interesting constructs (e.g., for extsyntax, or more recently for block structuring). These problems will persist, because the proposals need to maintain backward compatibility with the current standard. Similar problems exist when trying to get rid of obsolete features such as $NEXT and the naked indicator.

The MDC has chosen to address this by creating a new language separate from, and interoperable with, the language specified by X11.1. This action frees us from any notion of backward incompatibility, and provides a "clean slate" upon which to address these issues. By using this approach, current investments are protected, and future software developments can even ignore the new features. For those who need object orientation, either to communicate with "external" objects or within the language itself, a consistent model will be presented.

Object orientation leads to a different way of thinking, a different way of designing, and a different way of programming. This "paradigm shift" will be of more importance than the introduction of the language itself.

This language constitutes a considerable implementation challenge. All proposals must be sensitive to this challenge, and the authors of new proposals are encouraged to uphold the active participation of the implementors.

### 3. Strategic and Tactical Considerations

Numerous discussions within MDCC-E and SC15/TG2/WG1 have identified a number of features that should definitely be in the language, others that should definitely not be in the language, and issues that require further consideration. This resulted in the following lists (numbers within square brackets refer to votes [pro:con:abstain] within SC15/TG2/WG1 (d d 1996 09 28) on these features and issues):

#### 3.1 Features that must be included

- object orientation [not voted on]
- block structuring independent of eol [15:0:1]
- true IF-THEN-ELSE construct [14:0:2]
- CASE construct [12:0:6]
- different scoping of variables [15:1:2]
- X11.7 must be able to call X11.1 routine
- X11.1 routine must be able to invoke X11.7 [15:0:1]
- use OBJECTS instead of svns and ssvns [10:2:5]
- contain exception handling , error processing, and event handling [16:1:0]
- X11.7 should have optional variable declaration [10:1:6]
- allow only a single entry point per procedure or method [17:1:2]
- support for persistent data [18:0:1]
- support for persistent objects [15:0:3]
- retain some form of indirection [20:0:0]

#### 3.2 Features that must NOT be included

- naked indicator [13:2:2]
- partially specified areas such as VIEW command, BREAK command, $VIEW function where no standard semantics are defined [11:0:5]
- label+offset in control flow (such as DO and GOTO) [14:0:3]
- "peek" and "poke" support [10:2:4]
- $STORAGE
- mandatory variable declaration [9:3:5]
- reserved words [15:0:3]

#### 3.3 Other Issues

- How do X11.1 routine and X11.7 interoperate?

- It is not too complicated to define a calling mechanism between X11.1 and X11.7. However, their local (and global) variable concepts may be very different. It is not obvious if and how both languages operate on each other's variables.

- Since the X11.7 specifications are defined in a separate standard, the MDC might decide to shift to a different way of specifying the language syntax. Most notably, adhering to a YACC-like specification would make it possible that implementors obtain the changes in a useful (even machine-readable) form, allowing them to work directly from such specifications. This would open up the possibility of introducing a state-of-the-art alternative for the <transition diagrams> that were used in the past. Informal discussion with the implementors made clear that they do not expect to use such tools for their production releases. However, such an approach may help in other ways such as catching ambiguities and providing a (mental) model.

- The X11.1 commands, functions, svns, and ssvns must be carefully examined and their X11.7 behavior must be defined.

- The MDC will continue to investigate existing languages, and use their concepts wherever appropriate [8:0:10]

- X11.7 should focus on ease of learning [13:0:5]

- The significance of "white space" should be investigated (in one way or another) [10:8:X]

- Use of regular expressions instead of pattern match [4:2:11]

- Should X11.7 maintain left-to-right evaluation of expr? [8:5:7]

- Should X11.7 include postconditionals? [7:2:10]

- Should X11.7 retain the current form of indirection? [4:5:11]

- Include support for Transaction Processing is favored [10:0:6], but how (if at all) is this feasible when distributed databases are involved?

- Should X11.7 introduce multiple standard levels of conformance? [7:0:8]

- The functionality of the $TEST svn in the context of timeouts needs consideration [9:1:7]

- X11.7 should limit the application of the GOTO command [13:4:0]

*Arthur B. Smith chairs the MDC and is in charge of computer systems at the University of Missouri's Veterinary Medical Teaching Hospital. Email: art@vets.vetmed.missouri.edu*