Software Collection Project
at
The Computer History Museum

Bernard L. Peuto
Chairman SCC

Why we Are Here

- Software collecting and preservation is critical to preserving the history of the information age.
- The Software Collection Committee help fulfill the software mission of the Computer History Museum which is:
  - "To preserve and present for posterity the artifacts and story of the information age".
- Software collecting and preservation deal with many challenging and unresolved issues.
- Collecting and preserving software will involve and rely on many communities of interested parties.

Initial SCC Charter

- Decide what it means to collect software:
  - Artifact categories
  - Scope (what we do and don't do)
  - Priorities
- Focus on a web accessible archive.
- Pick few test cases and pursue them in depth.
- Establish policy and procedures for ongoing collecting activities.
- Get interested groups and individuals collaborating with us on this preservation initiative.
- Augment the resources needed to fulfill the Museum's mission of preserving software.

Computer History Museum's Software Collection

With thanks to
Kirsten Tashev
Director of Collections & Exhibitions

Samples by Type:

- Operating Systems: Multics, IBM OS/360, DOS, Windows (various vers.), Mac OS (various vers.), Pick, OS/2, Solaris, Free BSD, VMS, NextStep.

- Applications: PC and Mac office productivity apps, Apple II, Xerox Alto, CP/M apps, PC, Mac, Atari, Amiga, games, VisiCalc v.1 Beta, SpaceWar! Lotus 1-2-3 v.1.0, Internet Worm (Morris) source code.

- Tools: LISP, TX-0, PDP-X, C, FORTRAN, COBOL compilers for assorted mainframes and minis.

CHM Software Collection

Samples by Media:

- 400 reels of 9-T tape (mainframe + mini) [60GB]
- 400,000 punch cards [75MB]
- 3,000 5 ¼" floppy disks (PC software) [750MB]
- 500 spools of paper tape (TX-0, Whirlwind, DEC) [10MB]
- 10,000 pages of source code listings [10MB]
- 400 boxes of early PC/Mac software [200GB]
- **total: 275GB**

Draft Collecting Criteria

New Technology	New paradigm; e.g. visual programming for generating code or logic programming
New Users	Expanded the circle of existing users/created new users, e.g. BASIC, Windows
Raised the Bar	A definitive improvement on practice or quality
Advanced the field	Program or algorithm that moved the field forward, e.g. genetic algorithms
Interesting Failure	Never shipped, sold badly, or did not enjoy much usage, but was important, e.g. GEM
Seminal Inventor	Written by someone who contributed in an important way, e.g. Bill Joy
Unique	Custom solution to an interesting problem, e.g. Space shuttle control program
>25 years old	Collected due to rarity, e.g. JOSS

Preservation Steps

Rescue	Software Committee determines wish list and goes after it
Passivate	CHM receives and stores in accepted way, e.g. climate controlled
Conserve	Repair deficiencies in media, e.g. broken tape
Transcode	Pull bits off original medium
Archive	Receive files, assign metadata and store

Software Collection Committee's Work

SCC Work To Date

- SCC in numbers:
  - Started in September 2003
  - October 2003 workshop (co hosted by Grady Booch)
  - 10 meetings of the SCC to date
  - 16 plus attendees per meeting
  - 60 plus names on membership list
- Entering the third phase of our work:
  - First phase: standards and procedures
  - Second phase: test cases
  - Third phase: communities and digital archive

First Phase: Standards and Procedures

- Establish the metadata structures necessary for archiving our software
  - Adopting the Dublin Metadata as our foundation
    - Metadata defines items and relationships
    - Continuing to review our options
    - Will be tested while cataloging our software collection
- Develop software taxonomy
  - Selected UN Standard product and service code
    - http://www.unspsc.org/
  - More appropriate for specific programs & systems
  - Tested on Grady Booch's Classic Software list
    - (http://www.computerhistory.org/cgi-bin/wishlist.cgi)
  - Will be augmented by an exception list as we test it further
  - Manual & Cook Book being done

Describing Items in the Collection:

- Record the minimum amount of information necessary to uniquely identify an item and make it retrievable.
- Use one set of metadata elements to describe all type of items in the collection: hardware, software, documentation, ephemera, and medias.
- Use a standard set of elements to facilitate collaborative sharing among members of the research community, e.g. Dublin Core Metadata Element Set www.dublincore.org .

Sample Dublin Core Metadata:

- Title
- Creator
- Subject
- Description
- Publisher
- Contributor
- Date
- Resource Type
- Format
- Indentifier
- Source
- Language
- Relation
- Coverage
- Rights

- Each is a repeating field
- Only use those that are relevant
- Use controlled vocabulary
- Add fields as needed

Data Standards

Metadata:

- Title
- Creator
- Subject
- Description
- Publisher
- Contributor
- Date
- Resource Type
- Format
- Indentifier
- Source
- Language
- Relation
- Coverage
- Rights

Author, Designer, Photographer, Artist, Architect, etc.

Publisher, Manufacturer, etc.

Controlled Vocabulary
- Text
- Physical Object
- Image
- **Software**
- Sound
- Dataset
- Interactive
- Service
- Collection

Electronic Resources:

- Title
- Creator
- Subject
- Description
- Publisher
- Contributor
- Date
- Resource Type
- Format
- Identifier
- Source
- Language
- Relation
- Coverage
- Rights

Software System Requirements

- Make and model of the computer(s) on which resource is designed to run
- Amount of memory required
- Name of operating system
- Software requirements (including programming language)
- Kind and characteristics of any required or recommended peripherals
- Type of any required or recommended hardware modifications

Example

486/33MHz PC, Macintosh, or Power Macintosh; 8MB RAM; Windows 3.1 (or higher) or System 7.0.1 (or higher); Java-capable Web browser; VGA monitor

Physical Collection Taxonomy (as an example)

- *pre-computer*
- *calculator*
- *analog computer*
- *digital computer*
- *fixed-application digital computer*
- *game*
- *robot*
- *I/O*
- *memory*
- *interconnect*
- *component*
- *software*
- *test equipment*
- *ephemeron*
- *miscellaneous*

Taxonomy: Booch vs. UN SPSC

Booch	#	UN SPSC
operating system	40	Operating system
language	29	Compiler and decompiler
application	15	9 categories: Analytical, Business function specific, CAD, Flight control, GUI dev., …
game	15	Computer game; Action game; Adventure game
text editing	14	Word processing; Content authoring & editing
web	11	8 categories: Email, Info. exchange, Networking, Internet browser, Conferencing, …
scientific	6	Analytical or scientific; Expert system
database	5	Data management and query
AI	4	Expert system
-	2	Analytical or scientific; Compiler
algorithms	2	<none>
utility	2	Security and protection; Transaction security/virus protection
assembler	1	Development
cmd. & control	1	Industry specific
framework	1	Development environment
laboratory	1	Compiler and decompiler
music	1	Music or sound editing
video editing	1	Video creation and editing

Second Phase: Test Cases

- Determine a list of representative software to test our procedures:
  - Dominated by the availability of interested volunteers.
  - Focus also on cataloguing the existing software collection.

- Current projects:
  - Early Fortran
  - Algol W
  - 1401 Software
  - Multics
  - Page Description Languages -- Mac Paint
  - -- NLS/Augment
  - -- PDP-1 Software
  - -- Early CDC Software
  - -- Adventure

Early Fortran Work

- Fortran was one of the earliest high-level languages (designed 1954-1957)
- Its compiler pioneered many optimization techniques
- We've located compiler source code listings (paper and microfiche), and machine-readable source code on the Internet!

Early Fortran Work

- Acknowledgements:
  - Paul Pierce found and read the 7-track IBSYS source distribution tape
  - Rob Storey created an IBM 7094 emulator that appears capable of executing the compiler
- For more information, see:
  - http://www.mcjones.org/dustydecks/

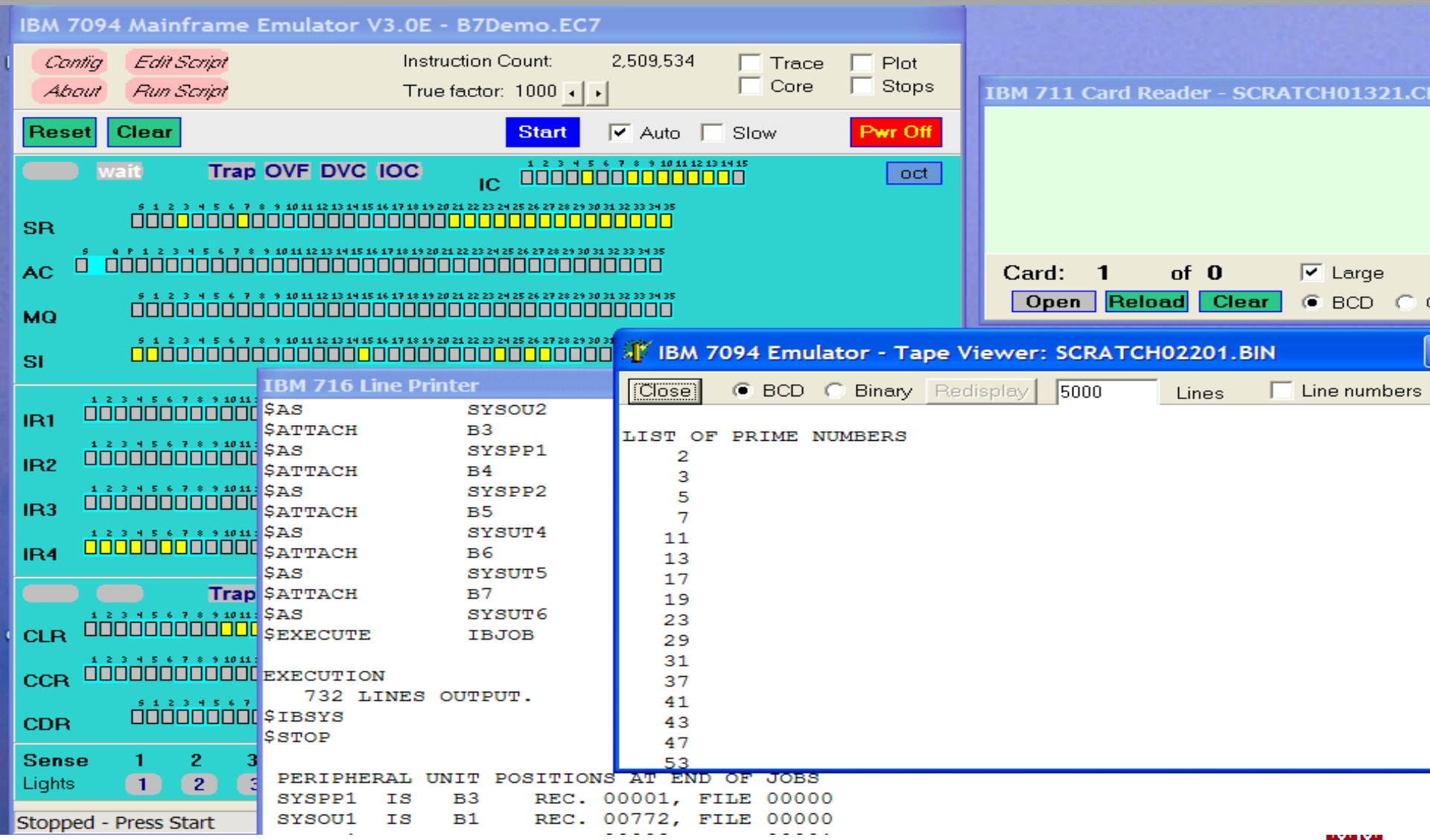
Early Fortran Work

```
        REM  PASS1 / CALLS ON RDCIT1 AND MIH                              F4A01200
        REM       THIS ROUTINE SCANS THE COMPILED INSTRUCTIONS AND        F4A01210
        REM  DIVIDES THE OBJECT PROGRAM INTO BASIC BLOCKS.  A BASIC BLOCK F4A01220
        REM  IS A STRETCH OF PROGRAM WITH ONLY ONE ENTRY POINT AND ONE    F4A01230
        REM  EXIT POINT.  THE OUTPUT OF THIS ROUTINE IS BBLIST, A TABLE   F4A01240
        REM  LISTING THE BEGINNING OF EACH BASIC BLOCK (THE LOCATION      F4A01250
        REM  SYMBOL OF THE 1ST INSTRUCTION OF EACH BASIC BLOCK).  A TABLE F4A01260
        REM  CALLED DOLIST IS ALSO COMPILED OF THE BEGINNING AND END OF   F4A01270
        REM  EVERY LOOP IN THE OBJECT PROGRAM.                            F4A01280
        REM       PASS1 ALSO STARTS READING TIFGO, TRAD AND FRET FROM     F4A01290
        REM  TAPE FOR LATER PROCESSING.                                   F4A01300
PASS1   ZAC                                                               F4A01310
        DCT                      BE SURE DIV-CHECK IND IS OFF.            F4A01320
        NOP                                                               F4A01330
        SXD     CITCNT,0         CLEAR DECREMENT OF CIT COUNT.            F4A01340
        LDQ     CITCNT           DIVIDE NO. OF WORDS IN CIT BY            F4A01350
        STQ     ILNGTH           LENGTH OF RESERVED AREA TO               F4A01360
        DVP     CTSPC1           FIND TOP OF LAST RECORD READ.            F4A01370
        DCT                                                               F4A01380
        TSX     ERRM4,IR4        DIVIDE ERROR GO TO DIAGNOSTIC.           F4A01390
        REM                           LENGTH OF CITS PLACED IN CITCNT     F4A01400
        REM                        BY SEC 3 , LENGTH OF CIT AREA DEFINED  F4A01410
        REM                           BY ASSEMBLY.                        F4A01420
        ADD     CTRD1            ADD BOTTOM OF AREA TO GET LAST LOADED    F4A01430
```

Early Fortran Work

```
        REM T/ TRANSFER TABLE (USED BY CF000).                    F1A17510
T       PZE     C0100,,2            DO.                           F1A17520
        PZE     C0200,,4            GO TO.                        F1A17530
        PZE     C0400,,14           IF (SENSE SWITCH.             F1A17540
        PZE     C0500,,13           IF (SENSE LIGHT.              F1A17550
        PZE     C0600,,13           IF DIVIDE CHECK.              F1A17560
        PZE     C0700,,21           IF ACCUMULATOR OVERFLOW.      F1A17570
        PZE     C0700,,18           IF QUOTIENT OVERFLOW.         F1A17580
        PZE     C0300,,0            IF.                           F1A17590
        PZE     C1000,,6            ASSIGN.                       F1A17600
        PZE     C1300,,4            STOP.                         F1A17610
        PZE     C0900,,5            PAUSE.                        F1A17620
        PZE     C1100,,10           SENSE LIGHT.                  F1A17630
        MZE     C1200,,9            DIMENSION.                    F1A17640
        MZE     C1500,,11           EQUIVALENCE.                  F1A17650
        MZE     C1400,,9            FREQUENCY.                    F1A17660
        PZE     C1600,,8            CONTINUE.                     F1A17670
        PZE     TSB,,8              READ TAPE.                    F1A17680
        PZE     TSH,,13             READ INPUT TAPE.              F1A17690
        PZE     DRS,,8              READ DRUM.                    F1A17700
        PZE     CSH,,4              READ.                         F1A17710
        PZE     STB,,9              WRITE TAPE.                   F1A17720
        PZE     STH,,15             WRITE OUTPUT TAPE.            F1A17730
        PZE     SDR,,9              WRITE DRUM.                   F1A17740
        PZE     SPH,,5              PRINT.                        F1A17750
```

Early Fortran Work



Mac Paint Work

Third Phase: Communities and Digital Archive

- Establish significant collaborations with a selected group of interested parties.
- Prototype the software digital archive.
- Finish the cataloguing our software collection
  - Heavily dependent on finding resources.
- An important Challenge:
  - Developing the web tools and procedures that would allow to open the committee to a larger group of volunteers.