# USING SOFTWARE DEVELOPMENT FACILITIES TO IMPROVE SOFTWARE QUALITY

S. J. TRENCANSKY, D. W. MEEHL AND H. C. ROMINE
THE SINGER COMPANY, LINK FLIGHT SIMULATION DIVISION
BINGHAMTON, N.Y.

A major problem exists in the development of current state-of-the-art weapons system trainers in the quality of the software provided. Since the software for a given trainer is typically generated on the deliverable computer system for the trainer a wide variance in the tools available to support the design and development of the software exists from project to project.

The specification requirement of a high order language (FORTRAN) has led to the evolution of software development facilities at various simulator manufacturers. While these have immediate impact in reducing the cost of the software produced and aid in maintaining schedule on the ongoing programs, their most significant impact is that the software delivered using them is significantly improved.

The paper explains what was experienced when the SDF concept was applied to some current contracts.

## 1. INTRODUCTION

The computer programs (software) developed for current state-of-the-art weapon system trainers (simulators) have been typically generated on the deliverable computer system, and thus have been subject to wide variance in the available software design, development, test and configuration control support programs (tools) from project to project.

Varying levels of available tools have been considered normal, especially when assembler languages were used on varying brands of computation equipment. The introduction of high-order languages (HOL), predominantly FORTRAN, as a specification requirement has led to the evolution of Software Development Facilities (SDF). The SDF concept was developed with three major objectives:

a. Reduce the cost of the software produced

b. Maintain or accelerate schedule performance of on-going projects

c. Maintain precise configuration control and management

These objectives have been fully realized, and the SDF provides deliverable software of significantly improved quality.

The Link* SDF has been designed to provide configuration control and management (CC/M) in a user friendly working atmosphere. Systems Engineering and software development personnel use conveniently located interactive terminals during all phases (design, development, test, maintenance, etc.) of unclassified projects.

*A trademark of The Singer Company

The principal features of this interactive capability are:

- *Prepare and edit source code*
- Compile and/or assemble the edited source code
- Prepare and edit Math Model Test (MMT) inputs
- Create a Software Change Request (SCR) for source code modified at the SDF

## 2. *SDF TOOLS*

Three basic categories of tools exist in all SDFs (see Figure 1):

a. Tools supplied by computer and software vendors

b. Tools developed by the software manufacturer specifically for the SDF, normally via IR&D programs

c. Tools developed under contract as deliverable contract end items

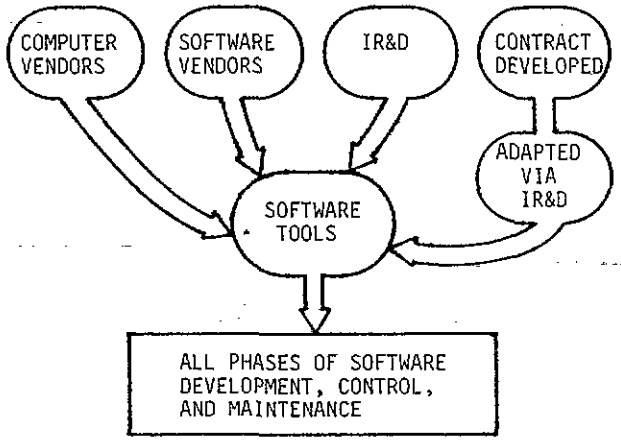


Figure 1
Software Development Facility Tools

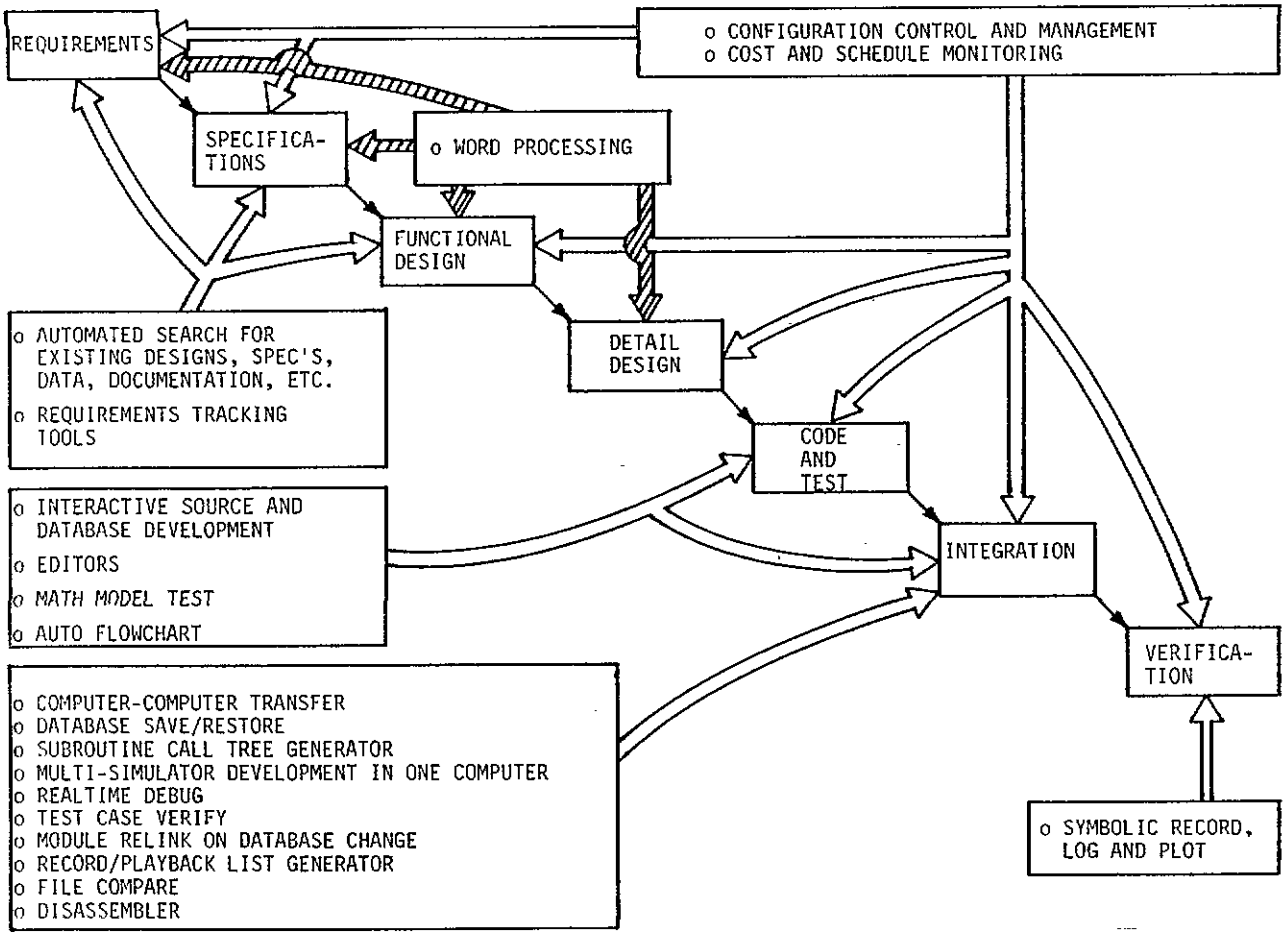


Figure 2 SDF Capabilities Used During Simulator Development

## 2.1 VENDOR SUPPLIED TOOLS

The Link* SDF is equipped with a large number of proprietary tools supplied by the computer vendor such as operating systems, data base management systems, terminal systems, compilers, assemblers, loaders, librarians, etc. The high-order languages (HOL's) supported include: FORTRAN, COBOL, BASIC, and PASCAL. In addition Link has purchased special purpose proprietary software for use during specific phases of a simulator's life cycle. Examples of this software include: a lens design package used by visual engineering during the proposal and conceptual design phases, a microcomputer cross assembler and emulator, sophisticated print and plot packages, a cost and schedule monitoring package providing critical path and other automated management capabilities and a comprehensive system providing full-field search and selected output of cited data within the Engineering Information System (EIS).

Figure 2 shows the traditional development cycle of a simulator through customer acceptance (verification). At each stage of this process, the SDF enhances our ability to quickly and accurately perform the indicated task.

## 2.2 LINK DEVELOPED TOOLS

Extensive systems and tools have been developed to control and enhance every access to the SDF. These capabilities may be best understood through an explanation of key features:

- o COMPILATION COMMANDS

  Single commands to accomplish the compilation, library edit, and production of various optional types of output, as discussed. Examples: ASSEMBLE, FORTRAN.

- o MULTI-SIMULATOR SUPPORT PACKAGE

  Software that interfaces to CC/M managed software. It basically allows copies to be made from CC/M disks, while prohibiting unauthorized changes. The copy can be used to develop the required version, along with the applicable database (Symbol Dictionary) entries applicable to the specific project.

- OPERATOR COMMAND STURCTURE/ VENDOR UTILITY COMMANDS

  Tools developed to utilize vendor software, formalizing the way is to be used by each operator. This is required to provide consistent operation.

- EDITOR/FULL SCREEN EDITOR

  Editors are used in developing software, test drivers, etc. These tools provide flexiblity, via English language commands, while protecting all fields used by CC/M software.

- FILE COMPARISON

  Allows automated generation of the CC/M required SCR by comparing configuration managed code to new code on a line by line basis. The printed SCR conforms with the specification required format.

- SPECIAL UTILITIES

  Single commands to accomplish repetitive functions (e.g., file copy, list, transfer, etc.). Also provides processors to inspect definitions of parameters and symbol edit.

- HELP

  Information capability, at the terminal, to aid users in proper SDF utilization.

- MATH MODEL TEST (MMT)

  A comprehensive software methodology that leads to consistent test and verification of design logic in a non-real-time mode of operation. Capable of exercising single modules, components, etc., up to an entire simulator load. Capable of accepting commands in either the interactive or command file mode. Software Quality Assurance requirements (e.g., test coverage reports, regression testing, etc.) are incorporated.

- PLOT PACKAGES

  Allows offline software and MMT to interface with various output devices.

- FILE TRANSFER

  Allows multiple computers to transfer files, in either direction, via RS-232 channels. Error checking/verification is provided.

## 3. IMPACT OF THE SDF

Before the SDF was developed, many tasks were performed on the deliverable computation equipment in a serial manner during software development and test. Now these same tasks may be accomplished in parallel on the SDF. This factor alone allows more online time for those tasks which cannot be performed at the SDF such as hardware-software integration. Figure 3 shows the serial versus parallel work flow.

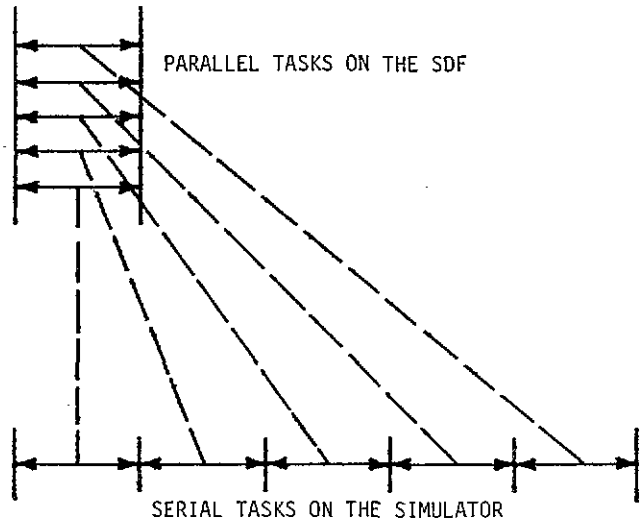


Figure 3 Serial Versus Parallel Work Flow

### 3.1 LOAD GENERATION

A simulator load may be defined as the complete set of executable software and data bases required to simulate the real world device, ready for insertion into the computer's memory system. Prior to the SDF, simulation loads were normally generated once a day. Because of the serial development, and limited available computer time for detailed testing, subtle problems existed with at least some of the newly incorporated software. Isolation and resolution of these problems reduced achievable progress in a given period of time. Furthermore, load generation and initial checkout, i.e., a cycling load, required an average of 2 to 3 hours. Figure 4 shows the process and results.

In the SDF environment simultaneous execution of tasks provides more time for each engineer to fully verify his module(s). Added testing, using MMT, yields software which is much more error free, thus reducing the load generation and checkout time to an average of 20 to 30 minutes.

Assuming one load per day, the use of a SDF provides an online savings of 1.67 to 2.5 hours per day, or from 51.8 to 77.5 hours per month (31 days).

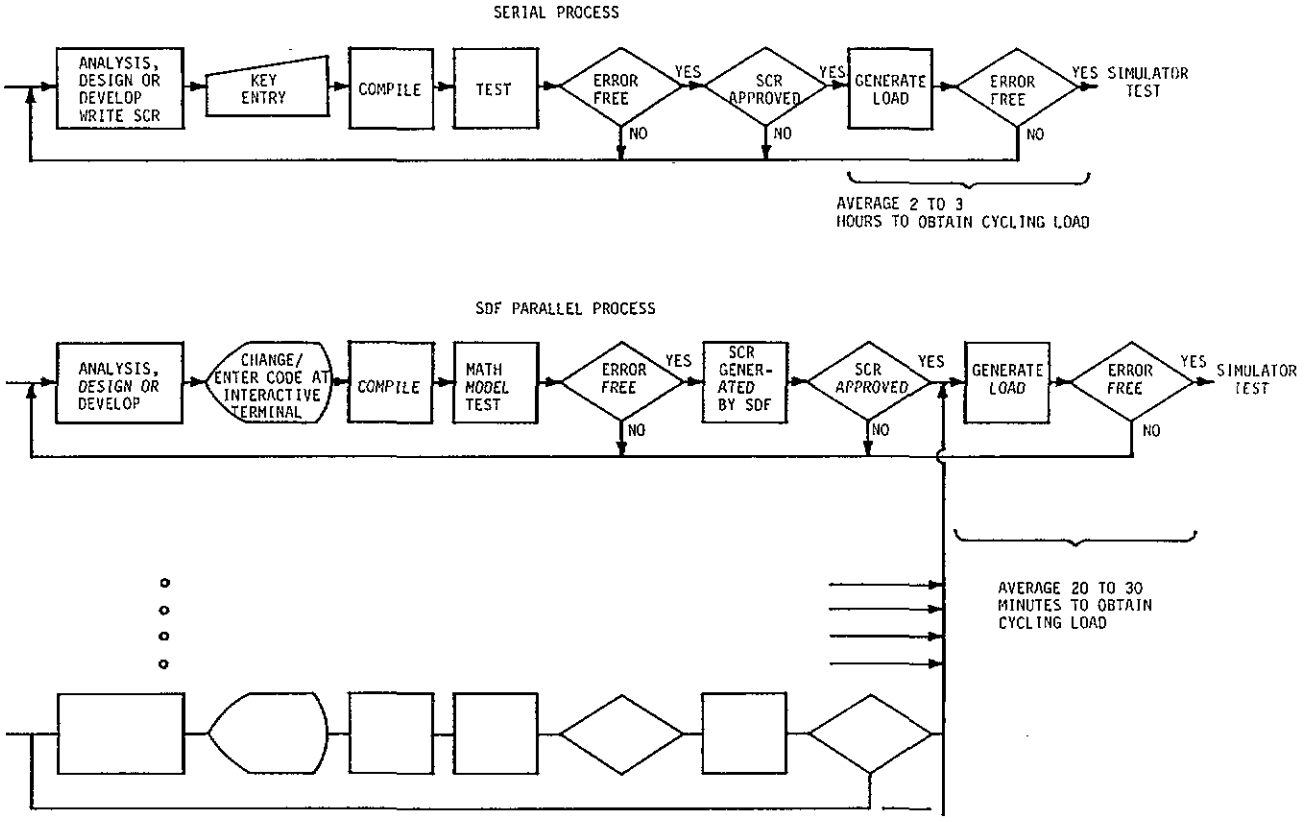


Figure 4 Comparison of Old and New Methodology On Load Generation Process

3.2 AIRCRAFT SYSTEMS

Aerodynamic and power plant (engine) simulation requires complex offline software to verify that the simulator design and implementation truly match the aircraft characteristics and performance as defined in the data available. The offline software also is used to develop usable test criteria in accordance with the specification. Before SDF, an offline computer complex was used to perform these functions. However, the ever-changing online computer vendors soon made it necessary to develop two identical sets of aerodynamic and engine modules; one in the language applicable to the online computer and the second in the language of the offline computer.

Use of the SDF allows significant savings, such as the following:

- o Acutal simulation modules are used in the SDF.

  Savings: No duplicate modules written for different computers; no duplicate CC/M costs.

- o Complex Test Drivers may be used across project boundaries.

  Savings: No duplicate modules or added CC/M; sophisticated drivers used on C-130, B-52 simulators.

- o Math Model Test (MMT) verifies design, implementation, interfaces, modifications, etc.

  Saving: More effective than traditional online debug; shortens online test phase.

- o Auto-Test Guide, an online test tools, is fully compatible with MMT, allowing identical checkout on the simulator and in the SDF.

  Savings: Reduced online regression testing associated with discrepancy clearance.

The SDF saved two weeks of online HSI time on the C-130 Program. The original HSI schedule provided three weeks for aero and engine integration, while the actual integration was completed in one week. Reason: More controlled development atmosphere, software quality controls, and in-depth testing on the SDF.

## 3.3 CONFIGURATION CONTROL AND MANAGEMENT

The SDF incorporates many features which assure that CC/M is correctly accomplished.

- o Fully controlled access to all levels of software.
- o Added controls limit controlled software updates.
- o Log of all users, the functions performed, date, time, computer resources utilized, etc.
- o Automated revision logging on target module and in CC/M database.
- o Automated SCR generation, log and reporting.

In addition to automated capabilities, manual intervention features have been incorporated into the SDF, and are used by CC/M personnel on projects such as the B-52 WST. Example of these functions, relative to incorporation of a B-52 WST change, are:

a. Determination of applicability, i.e., All B-52s, B-52G only, B-52H only, etc.

b. Determination if a basic (used simultaneously in more than one station) module.

c. Determination if the change must be retrofitted to previously delivered simulators.

d. Verification that all applicable documents have been updated and that copies are in the change package.

e. Verification that software quality requirements have been met.

When these actions are completed, the complete package is presented to the Software Change Control Board for approval or disapproval. Approved changes are logged into the CC/M system and then sent to the Software Controller(s) for incorporation into the load.

Historical data, current status, module effectivity listings, software trees and reports on all activities are available from the SDF upon request.

Cost savings (avoidance) has been estimated at $20,000 per month by using the SDF for CC/M. This figure represents our estimates of additional personnel required in lieu of the SDF for CC/M.

## 4. DEPS CAPABILITIES

Many current contracts include an offline computer complex designed to support the simulator at the customer's installation. The similarity of mission in the SDF and DEPS indicates a potential for transfer of SDF software to the DEPS.

## 5. CONCLUSIONS

Usefulness of the SDF has been proven at Link. A recent U.S. Air Force simulator contract had a program plan, developed prior to the SDF, which contained a six month Hardware-Software Integration (HSI) schedule. While the contract was in progress the SDF became operational and was fully used to control, develop and test software. The effect of the SDF is shown in Figure 5.

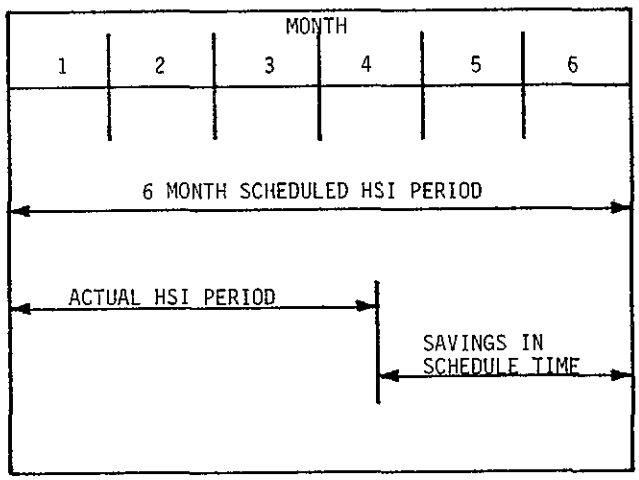


Figure 5 SDF Effect Upon Major Simulator

During the software test phase (pre-HSI) the SDF recorded 3400 hours of use. Since the types of testing being done on the SDF were identical to the testing normally done online as pre-HSI test, one may conclude that an hour on the SDF saved an hour on the simulator. Further, assuming that the simulator was available for test 16 hours per day and 31 days per month, the 3400 hours translates to 6 months of simulator time.

HSI on this simulator progressed unusually well with very few module interface problems. This resulted in the HSI phase being completed in about one-half of the time originally scheduled. Each simulator manufacturer and customer can calculate their estimated savings when the schedule is reduced by 2.8 months.

What will the future reveal? Certainly more complex and difficult requirements and specifications, especially in the areas of verification and validation. Will today's SDF and tools meet future needs? Not totally, but proven SDF philosophies, designs, procedures, tools, etc. will give the SDF owner a competitive edge.

## ABOUT THE AUTHORS

Mr. Hugh C. Romine is Engineering Director for Government Simulation Systems - Binghamton Operations at the Link Division of the Singer Company. He received a B.S. Degree in Computer Sciences at Florida Technical College and has post-graduate work at SUNY-Binghamton.

Mr. Stephen J. Trencansky is Staff Scientist for engineering development, Government Simulation Systems - Binghamton Operations, on the Engineering Director's Staff at the Link Division of the Singer Company. He received his B.S. Degree in Physics from Clarkson College of Technology and M.S. Degree in Computer Sciences and Applied Mathematics at SUNY-Binghamton.

Mr. Dennis W. Meehl is Staff Engineer and principle investigator for the Software Development Facility IR and D project at the Link Division of the Singer Company. He received his B.S. Degree in Engineering at Case Western Reserve University.