Standardized Systems Development

by

Joel Robinson
Manager
Information Systems
Wimpey Canada Limited
Toronto, Ontario,
Canada

1.0 INTRODUCTION

Imagine, if you will, an 80 million dollar company with operations in both the U.S.A. and Canada sending every one of its transactions on paper to its head office in Toronto for processing. The processing was done on a single magnetic ledger card bookkeeping machine. Imagine the chaos, the delays and sheer volumes, for Wimpey is a large construction and real estate development company. This was the situation when I was hired in the summer of 1976 to study the company's paperwork! By the summer of 1977, the company had installed an HP-3000 and I began to search for that most exclusive of beasts, The Programmer.

In the period from July 1977 to January 1979, the newly formed Information Systems Department (I.S.D.) turned out a range of commercial systems using Image, COBOL and DEL, with full time manpower on these projects of three people. The only way one could succeed in developing custom systems was to force a rigid development approach on all systems developers. This paper outlines some of these approaches.

Presently Wimpey I.S.D. has a staff of eight people, two HP-3000's (a III and a 30), and terminals across North America. We have a combination of leased line, packet switching (Datapac and Telenet) and hardwired circuits to support our 40 terminals which are primarily HP-2645's and 2635's. We have a wide range of online technical and services systems, developed in FORTRAN, BASIC and KSAM for our various businesses and these are outside the scope of this paper.

2.0 STANDARD USER INTERFACE

With users of various types and spread across North America, the word "idiot proof" took new meaning. Users would need to be guided through any operation and would be expected to not only make numerous mistakes but also not to have even referenced any supplied documentation. This was proven to be the case, repeatedly. Logging in could take some users up to two minutes.

Hence, block mode and menus were chosen as the best way. Later MPE allowed us to automatically run the system via a logon UDC, further simplifying the interface. So far, nothing new. But these standards were developed and consistently implemented in all systems (some of which are obvious and some of which VIEW can do for you):

1. All input fields are inverse video/underlined.

2. All display only or title fields are inverse video/half bright.

3. The first box is always the mode:
   A - Add
   C - Change
   D - Display
   P - Purge
   R - Recover
   X - Exit

4. The error line/action line is always at row 24.

5. Errors or action messages appear but an action (pushing F1 to F4) is required to acknowledge. Explicit confirmation reduces errors. The use of the buttons is consistent.

6. Functions F5 to F8 are local cursor positioning functions with F5 on display being used for hardcopy via an attached hardcopy unit if present for display only users.

7. Security is implemented via the local attribute and the menu initiates all functions after a security check. Lockwords and capabilities reduce the chance of users running programs directly. Key security is also checked within critical applications programs as an added check. The nature of our business is such that security cannot be overlooked. The opportunity for large cash frauds must be minimized.

3.0 STANDARD DATABASE DESIGN

There are no major breakthroughs here but in two years we have never lost any data. All master maintenance is direct with no logging, other than modification/create dates. All transaction entry is indexed by a batch control number. Thus, if a user goes down, logging in again and "recovering" the batch will enable one to complete partial transactions and continue on correctly. Extensive logging is not required.

By using batch control, auditors can be satisfied that no modifications take place after entry and any question of database integrity can be readily proven at any time. As a policy, no transactions are ever removed during their current fiscal year.

4.0 STANDARD FUNCTIONAL MODULES

Too many programmers spend time defining program types but this has been eliminated via our approach. Through extensive use of boilerplate, cut and paste, and common subroutines, we have built our systems faster than conventional approaches. The main functional types are :

1.	Menu	-there is only one -it creates and activates the function selected as a process -it checks security
2.	Master Maintenance	-simple, straightforward -usually shows transactions via second screen for online inquiry -workhorse programs
3.	Transaction Entry	-batch controlled -actions include add, resubmit, force, delete batch -controls tight
4.	Initiators	-simple screens which users fill in to create jobs run in batch
5.	Batch Runs	-anything which updates critical information -exclusive access with tape backup prior
6.	Reports	-sometimes normal COBOL -sometimes dynamic QUERY/ASK where initiator creates the select and adds parameters to the report procedures

The bulk of these modules are in COBOL, except for some work in BASIC. A standard number scheme is followed and structured methods are employed. Rather than depending on a standards manual which would never be referenced, we tend to copy and cut a working module in another system for a new system. Sometimes a skeleton is used alternately.

5.0 BENEFITS

The benefits of the standard are apparent but continuing improvements are being made. Now VIEW has simplified our screen development and transaction logging may help us in the future but for now the approach has enabled us to :

1. Install systems in a wide variety of locations.

2. Quickly modify systems for newly acquired companies.

3. Develop these systems with 3 people in 18 months :

   - General Ledger with certain subsystems
   - Payroll hourly, weekly and monthly with EFTS
   - Accounts Payable
   - Physical Plant Utilization System
   - Corporate Mailing List

4. Develop in six months with 1 1/2 people :

   - Invoicing and Sales Analysis
   - Accounts Receivable

5. Allow maintenance efforts to be done readily by anyone without undue learning effort.

6. Allow priority effort towards database definition and functional feature analysis and less towards functional analysis.

7. Pass numerous audits, internal and external.

8. Develop without expensive and difficult to manage project leaders or systems analysts.