

FAST Development Plan Proposal
By Chandler Gabel, Life Cycle Engineering
Last updated 28 August 2002

Overview

The FAST development cycle can be divided into five logical steps:

1. Selection of development tools
 - Database system
 - Reporting software
2. Definition of database structure
 - Table definitions
 - Field definitions
3. Creation of software engine
 - Data translation routines
 - Common code libraries
4. Creation of user interface
 - 2-Kilo/CSMP/ICMP editor with integrated support for parts
 - SCLISIS editor
 - MRC editor
 - Visit & ship information editor
 - Data translation interface
5. Testing and Improvement

This proposed development plan explains each of these steps in detail.

Selection of Development Tools

Although the FAST codebase itself will be developed in Microsoft Visual C# .Net, the selections of several ancilliary development tools have yet to be made. Among these are:

- The database system in which the tables and queries related to the ship visit data will be defined. Microsoft SQL Server and Oracle are the most widely used database systems currently on the market.
- The reporting software that will be used to create the user-friendly presentations currently produced by Matrix97 and C2Reports. Microsoft's Visual Studio .Net includes a license for Crystal Reports v8.0; this well-known extension could provide integrated reporting capability like that currently implemented in PVAT.

The selection of development tools should be based on a list of program requirements, so such a list must be compiled before this step can be carried out.

Recommendations: The basic structure of FAST is one in which information is downloaded from a central web database to individual desktop machines, in keeping with the NMCI standard. Keith Conway recommends that this central database be created in Microsoft SQL Server 2000, and that information be stored on individual machines in Microsoft Access format.

Definition of Database Structure

Before any code is written, a coherent database structure must be defined. This process will involve both the Norfolk and San Diego offices, since the goal of this step is to design a structure that combines the capabilities of the current PVAT and C2 databases with a consistent and modular design amenable to future product extensions, and will involve the following steps:

1. Using the program requirements as a guide, define the structure of the database as a set of tables (for instance, it might make sense to store 2-Kilo information in one table, SCLISIS data in another, etc.), taking into account the ways in which the various tables must interact with one another and the consideration that the structure must function as a basis for both the central database on the Internet and the individual MS Access databases on end-user machines.
2. Define the structure of each individual table. This will involve two steps: first, determining the information that each table must store (a 2-Kilo table, for instance, should include a work center and a job sequence number); and second, representing that information as a set of data fields.

Creation of Software Engine

Once the database structure has been defined and a sample database created, the coding of the software engine – the underlying routines, transparent to the end user, which drive the program’s behaviour – can begin. The FAST software engine will consist of two major components:

1. Data translation routine(s). One of the major requirements of the FAST software is that it be able to export and import data to and from several existing formats (PVAT databases, ICAS data, MM0001 SNAP upload disks, etc.) Writing the code to implement this capability should be one of the first steps in developing the FAST engine, since when finished it will enable the testing of FAST with actual visit data. An ideal solution would involve a single translation routine which could be configured to read the formats mentioned above and any others which FAST might be called upon to read in the future.
2. Common code libraries. Since FAST is the first C# project to be developed by LCE, there is no internal library of reusable code on which to draw. Previous

projects written in Visual Basic 6.0 made extensive use of code libraries written by the LCE staff; porting the most relevant of these to C# will allow FAST to take advantage of functionality which has already been tested, refined, and proven.

Creation of User Interface

The most significant step in the FAST development process is the design and implementation of the user interface. This phase should include regular feedback from the end users, since it is important to ensure that the interface design is in keeping with their requirements. The finished FAST user interface must include the following components:

1. A 2-Kilo editor with integrated support for parts tracking.
2. CSMP and ICMP data editors. LCE Norfolk's work on the CSMPITS 2000 (C2) project has shown that this functionality could be integrated into the 2-Kilo editor, thereby decreasing development time and the size of the project code.
3. A SCLISIS data editor. Both C2 and PVAT include this functionality, and both would serve as good models for FAST's SCLISIS editor.
4. An MRC editor, as seen in PVAT.
5. A ship and visit information editor, where end users could enter such information as the name of the ship they are visiting, the type of visit they are conducting, and the dates that the visit is scheduled to begin and end.
6. A user-friendly interface for the data translation functions (see "Creation of Software Engine").
7. A configuration screen in which users could tailor the operation of FAST to suit their personal preferences.

Testing and Improvement

External beta testing of FAST can begin as soon as a functional user interface is in place. Ideally, members of the FAST team should accompany end users during ship visits to offer real-time support for the program and to listen to suggestions for improvement. Once FAST is both stable and well-suited to the field requirements of the end users, it can be placed into widespread use and further refined in light of the resulting feedback.