The Synergy Approach™

Raghav S. Nandyal
Chief Executive Officer

SITARA Technologies Pvt. Ltd.
'Bellevue', #559 • Road No. 3 • Banjara Hills • Hyderabad AP - 500 034 • INDIA
Email: raghav_nandyal@SITARATECH.com
URL: http://www.SITARATECH.com

A Near Zero-Defect Approach to Software Development

From The

Software Engineering Laboratory

March 1995

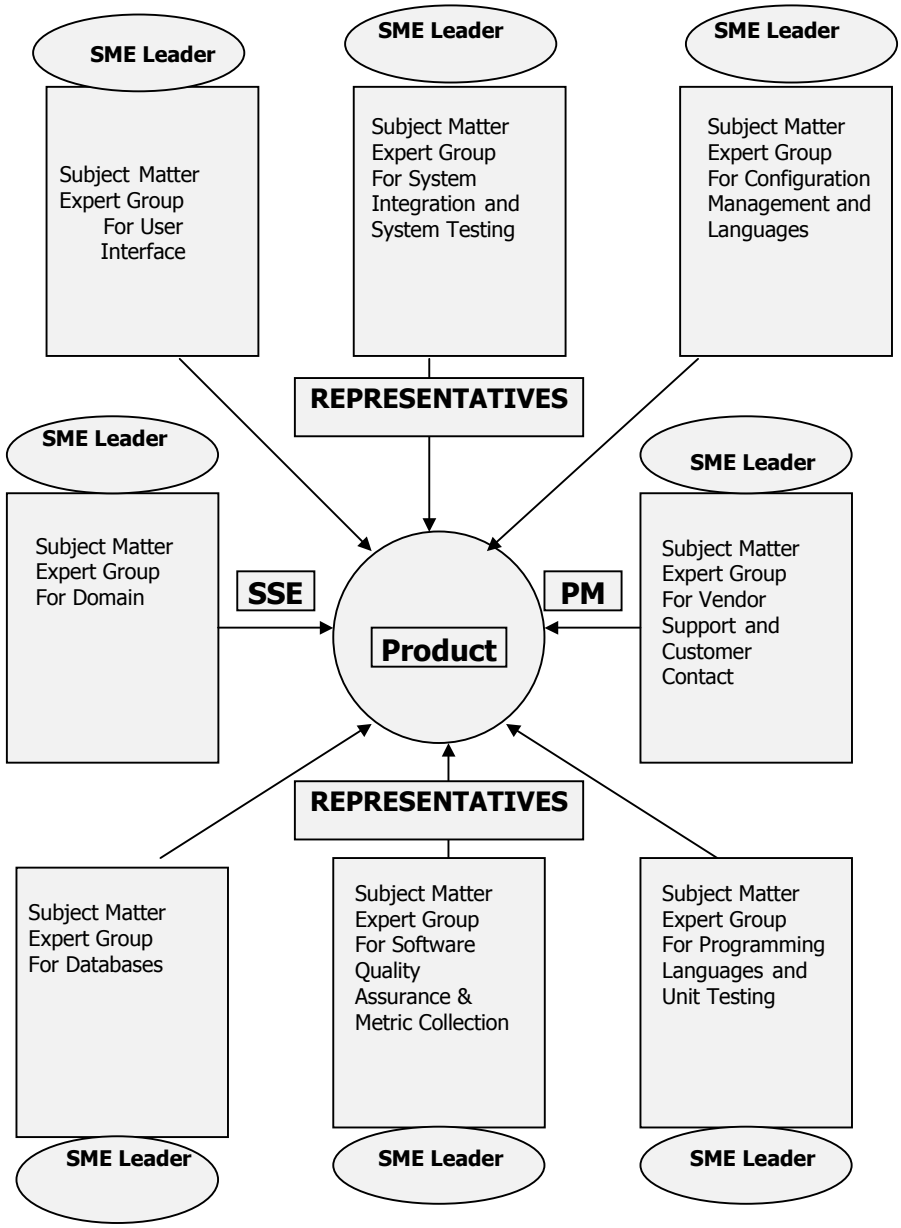
2.0.1 Technical Pool organization

Some of the key areas in which software development effort will require expertise are the following:

- Graphical User Interfaces
- Configuration Management and Computer Languages
- Software Tool Support and Network Administration
- System Integration and System Testing
- Software Quality Assurance and Metric Collection
- Databases
- Domain
- Vendor Support Information and In-house tool development
- Programming Languages and Unit Testing

The idea behind this organization is to build in-house subject matter experts (**SME**) who will work interdependently to provide software solutions with the following project organization structure.

2.0.2 Project Organization Structure



2.1 Requirements Definition Phase

This is the phase wherein the management and the representatives from the technical pool who will assist in making a feasibility study evaluate the customer's wish list. The nature of activities in this phase will involve project staffing and project schedule determination. Identification of the right staff profile and the training that will have to be imparted to the staff is evaluated in this phase. The configuration item or the end of phase deliverable from this phase would be the Functional Requirements Specification or **FRS**. The corresponding System Acceptance Specifications **SAS** can be drawn up in order to form inputs to the software system testing activities.

Some of the important points to be emphasized in the FRS, are - *what* the system will do rather than *how* it is to be done. What is more important to be recognized and documented is *what the system will NOT do*. Oftentimes, a failure to recognize these implicit requirements will require rework during the design phase causing to increase the cycle time of the system development.

The focus of this phase is on "functionality" that would have to be realized given the "performance objectives" that would have to be achieved under known "constraints of operation" based on the "data or information flow and content". This system definition will lead the software systems engineer (**SSE**) to collaborate with the subject matter experts (**SME**) who will be assigned to the project to come up with the Function Requirements Specification. The FRS document is a "logical and functional" break up of the overall system behavior. These logical or functional units will state the objectives that would be accomplished from a functional unit in as detailed a manner as possible. These logical units will then be decomposed into sub-requirements which when put together will accomplish the overall requirement of the unit. When this process is continued to encompass the entire system, the product functional requirements specification can be drawn up. This will result in the requirements book or **ReqB** or the **FRS document** as the configuration item or the end of phase deliverable. Together with the FRS document, inputs to the project management plan to come up with the project schedules will lead to the Software Project Management Plan (**SPMP**). SPMP will of course undergo modification in the requirements analysis phase to make more realistic project deadlines. Basically, the SPMP at the end of requirements phase will detail the need to have a phased delivery of the product or whether the project is do-able within the time constraints. The SPMP will also make an assessment of whether the resources – manpower, money and tools - to execute the project are available.

It is important to identify customer needs from the customer wants for better prioritization.

NOTE

The example discussed here is NOT complete nor is intended to be a complete solution. It is NOT in the recommended ReqB or FRS format since the intention of this example is purely for illustration purposes.

Example:

Problem Statement: It is required to build a real-time clock featuring the display of time.

ReqB or FRS:

R1. Real-time clock that will display the time using the system clock as initial input has to be developed.

R2. The display of time can be either in the 24 hour format or AM/PM format.

R3. The time can be set to any valid value after the system initializes itself to the system time.

R3.1 Validation of user-selected time must be done in both the formats.

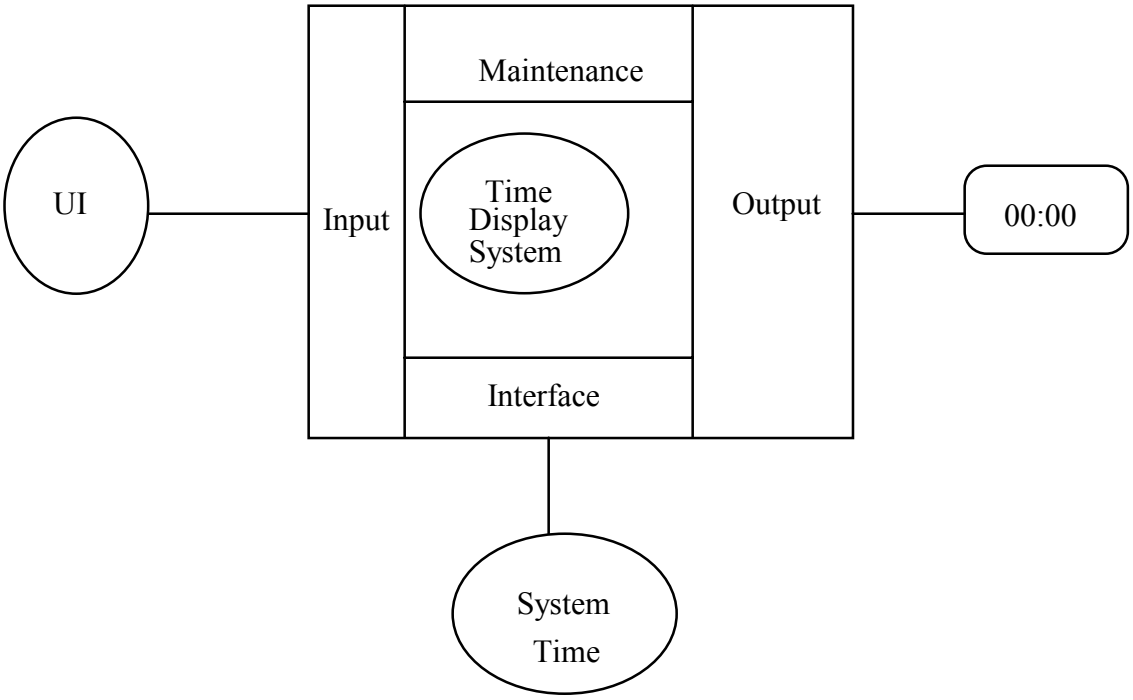
R3.2 Invalid time must prompt the user to correct the mistake.

R4. The display shall update itself at the end of every minute.

R5. The display format shall be as HH:MM AM/PM in the AM/PM format.

R6. The display format shall be HH:MM in the 24 hour format.

The FRS will also have a system interconnection/interface diagram in the following format.



Time Display System	:	This is the software solution that has to be built.
Interface	:	This is the system interface to other processes.
Maintenance	:	This is the interface available for tuning the Time Display System.
Input	:	User access to the Time Display System.
Output	:	This interface will display the output to the user.
UI	:	User inputs to modify the time.

It is very clear from this diagram that, what will have to be developed is the Time Display System. This diagram also gives insights into the hooks that would have to be designed to get the system time and also the formatting that would be necessary to perform the display. This diagram also suggests that necessary hooks to receive user inputs must be designed as well.

The deliverables will have an impact from the customer-imposed standards for Software Development, Design Standards, Design Representation, Coding Standards and Testing Standards. So, it is important for the requirements gathering team to recognize these

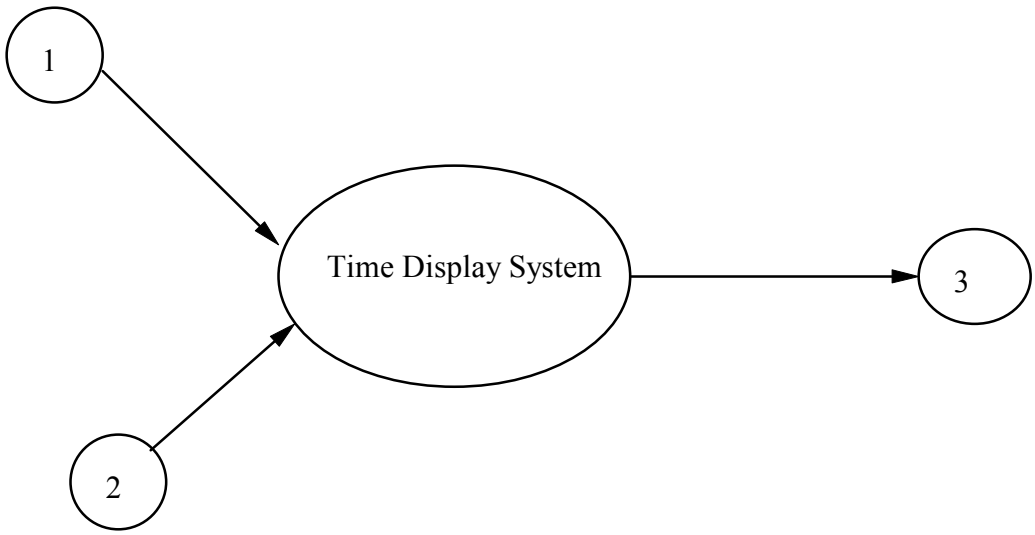
standards. Quite often, when such a feasibility study is undertaken, there will be technical and non-technical risks involved. Suitable abatement procedures must be well thought out.

2.2 Requirements Analysis and High Level Design Phase

Inputs to this phase are primarily the requirements book or ReqB. Other inputs could be domain literature and users manual by the customer. If there are customer-imposed tools for development, then tool related documents such as user's guide and reference guides, source code and integrating strategies form a very useful input to this phase. Two very important inputs to this phase are - what I like to call the "**Red Book**" and the "**White Book**". A Red Book is a book detailing the mistakes committed in the past on a similar project honing all the "error alerts" from first hand experience. This serves three purposes essentially: avoid repeating the same error, publicize a positive quality service and finally doing the right thing right the first time. A White Book is more of a domain specific in nature that documents issues which were either overlooked or were done differently on a similar project, but the next time over it would be done right and would not be repeated. These two books are developed in the Code Standardization and Unit Testing phase and will be continued all the way up to the customer approval phase. This is to document the quite often perceived feeling -"... had I done it like so and so, things would have been entirely different! ***This,*** is the problem with the design!". More on these books will be elaborated in the Code Standardization and Unit Testing phase.

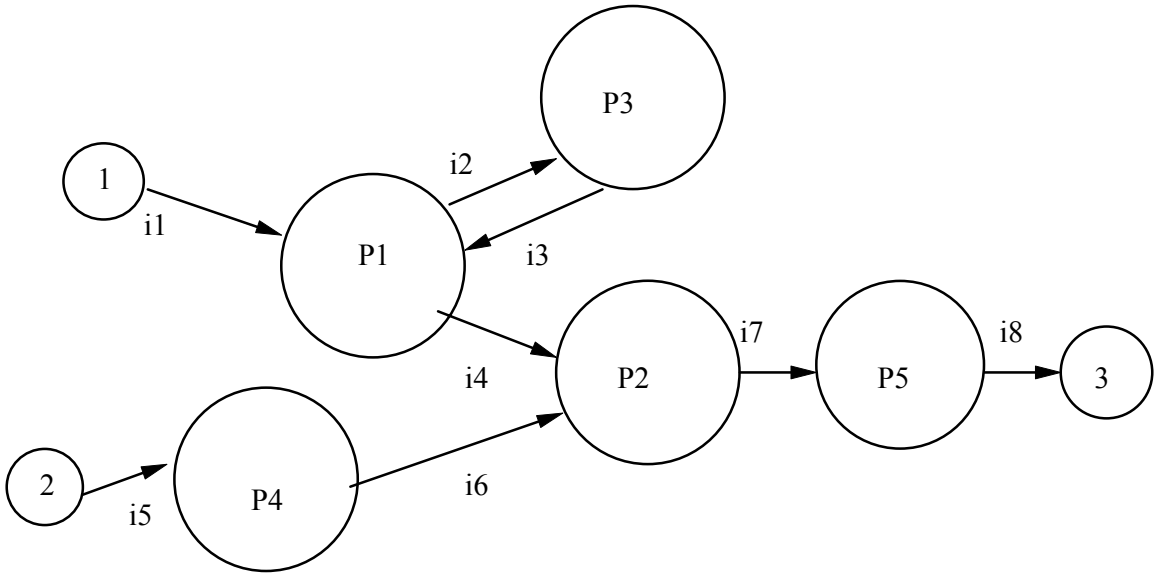
The Requirements Analysis phase will commit the requirements into "processes or process bubbles". A process is a function or a collection of functions that would accomplish a given requirement and move the overall software system state to a level closer to the end result. This method of evolution introduces a top-down solution to a software problem.

The very first bubble will be the "Time Display System". This bubble will receive the system time at initialization. It could also receive user modifications subsequently and then, it will send the output for display. So, this bubble or process is called process 0 of DFD 0 (Data Flow Diagram 0). And, it will look like:



1 : User input to the system for modification of time.
2 : System Input providing the system time.
3 : Output from the Time Display System displaying the time.

Now, we work our way down from this top-level bubble or process. This bubble can be decomposed into a "validate time" process or bubble, a "format time" process, "update time" process, "error handler" process and a "display time" process. This of course is a very broad process allocation. There may be sub processes that may have to form a layer of support that could be system/tool dependent. Without getting into the details, the purpose here is to merely highlight the organization of the solution.



Time Display System

P1 : Validate Time Process
P2 : Update Time
P3 : Error Handler Process
P4 : Get System Time Process
P5 : Format Time and Display Time Processes
i1 : User input
i2 : User input string
i3 : Validation status
i4 : Validated string
i5 : System time
i6 : System time for update
i7 : Updated time for display
i8 : Formatted time string for display

Each of the above processes will have a corresponding process specification (PSpec) that will describe the functionality that will be accomplished by the process or will further be decomposed into processes. What is important is data balancing. Corresponding to each one of these processes, there should a traceability established to the original requirements. So, a traceability matrix comprising the requirement to which

the process traces back must be generated. The format in which the PSpec will have to be written must consist of the following -

TITLE :
INPUTS :
OUTPUTS :

PROCESSING :

And, a sample PSpec for Validate Time Process could be :

TITLE : Validate Time
INPUTS : Time
OUTPUTS : Validated Time Status
PROCESSING :

Determine the input time format.
Check if the time is valid in the format that the time has to be displayed.
IF the time has a valid format, THEN return a VALID status. ELSE return an INVALID status.

It is important to note that, the PSpec merely describes the processing without getting into details of "how" the processing needs to be done and what the variables/flags are and the values they will be set to.

The end of phase deliverables are:

1. The Data Flow Diagrams, Control Flow Diagrams
2. Process Specifications (PSpec)
3. Interface Specifications (ISpec)
4. Data Dictionary

These form the inputs to the next phase.

END OF LECTURE 2

CONTINUED ON LECTURE 3 [SITARA SE/JULY 2001]