



Republic of the Philippines
Department of Social Welfare and Development
Batasan Pambansa Complex, Constitution Hills
Quezon City
Telephone No. 931-8101 to 07

Memorandum Circular No. 01
Series of 2005

SUBJECT: INFORMATION SYSTEMS DEVELOPMENT GUIDELINES*

1. RATIONALE

Systems are developed to solve problems and/or enhance activities in an organized way. Its development mainly deals with careful analysis and design of new procedures.

Computer-based systems are now developed to fast track and ease management decision-making. Everybody wants to take advantage of the technology resulting to a great demand for Systems Development. Since DSWD does not have any written standards, systems development efforts initiatives are disorganized, costly and not well planned. There is no clear role relationship among users, developers and owners.

Further, the most critical barrier in successful software development projects in the public sector is the lack of Standards, Formal Processes for software project development and acquisition which include guidelines on requirements definitions, evaluation, quality assurance and final acceptance of software projects.

Thus, to ensure the efficiency of the systems developed in and for the Department, these guidelines have been created.

2. OBJECTIVE

The intent of these guidelines is to:

- Establish standards and procedures in developing and/or acquiring Information Systems;
- Ensure that developed/acquired systems will be both effective and efficient based on functionality and cost-effectiveness; and
- Provide clear understanding of the roles that different stakeholders (Management, Users, Developers, MISS, etc.) have in the development/acquisition, implementation and maintenance of systems.

3. LEGAL BASIS

- 3.1 Executive Order 265 - Approving and Adopting the Government Information Systems Plan (GISP) as Framework and Guide for all Computerization Efforts in Government.
- 3.2 DSWD Memorandum Order No. 30 (Series of 2003) - Constituting the Management Information System Service of DSWD.

- 3.3 Republic Act No. 8792 (Electronic Commerce Act) - An Act Providing for the Recognition and Use of Electronic Commercial and Non-Commercial Transactions and Documents, Penalties for Unlawful Use Thereof and for Other Purposes
- 3.4 NCC Memorandum Circular 2004-01 - Repealing NCC Memorandum Circular No. 99-02 "Prescribing Guidelines For I.T. Resource Acquisition in Government" and Providing New Guidelines Therefor
- 3.5 DSWD Information Systems Plan 2002-2006
- 3.6 Commission on Audit Circular No. 97-003 - Accounting Guidelines on the Acquisition, Maintenance and Disposition of Information Technology Resources

4. DEFINITION OF TERMS

The Definition of Terms found in Annex A shall be used, and shall form an integral part of this Circular. The Definition of Terms may be updated from time to time to reflect new hardware, software, services, and new perspectives in the field of Information Technology particularly the Internet.

5. SYSTEM DEVELOPMENT STANDARD

5.1 Key Concepts

5.1.1 Project Type. Project type refers to the characteristics of the project. In this context, the *DSWD* software project types can be categorized as:

- **New Systems Development Project:** development of new software. This typically involves automation of an existing manual business procedure.
- **Systems Enhancement Project:** major modification of function, feature, performance or interfaces of existing software. This project may use the same environment of the current system.
- **Systems Re-engineering Project:** project that involves restructuring of data, architecture and logic. Restructuring is performed to produce a design that does the same function better than the original program.
- **Conversion Projects:** change existing system to a different format, platform or environment with no change in functionality. This may be done to keep up with latest technology.
- **Systems Acquisition:** purchase of software which can be customized to fit the needs of the Agency.

5.1.2 Project Development Team. The project team is a very important component of any software development project. Every *DSWD* software project development team must, at the minimum, be composed of:

5.1.2.1 Project Manager who shall oversee the entire project development process to include:

- Ensuring that the project and the deliverables or outputs are on schedule;
- Supervision of team members;
- Coordination of efforts among user community and development team.

5.1.2.2 Quality Assurance Team who shall ensure that the software meets the requirements and quality standards. The team shall perform the following activities:

- Application of technical methods,
- Conduct technical reviews,
- Software testing,
- Enforcement of software standard.

5.1.2.3 Systems Development Team composed of:

- **System Analysts**, who shall coordinate, gather, refine and prioritize the various user requirements and develop the design specifications.
- **Programmers** who shall translate the design into codes.
- **Documentation Analysts** who shall be responsible for producing user documentation and assist in the development documentation.

5.1.2.4 Systems Support and Maintenance Team who shall be involved in maintenance of existing software that includes activities such as:

- Customization
- Software maintenance
- Installation

5.1.2.5 Training Team who shall formulate systems training syllabus and provide usability training to clients.

Annex B illustrates a typical organizational structure of a Project Development Team.

5.2 Methodology

A well-defined methodology is a key success factor in software development and an important instrument in controlling resource management, design and quality assurance. Software methodology is composed of a software development model used in conjunction with one or more techniques and tools.

It is important to structure the development of software to:

- Have a standard approach in systems development.

- Provide guidelines to follow for every activity in software development and the sequence they are carried out.
- Make project planning and management easier.

5.2.1 Software Lifecycle Model

For both in-house and outsourced new systems development project, the Agency shall implement the methodology for open source or open software development. It is a more modern approach to software development that is similar in parts to widely known software development methods. The major benefits of open source model include:

- Participation in global network of software development
- Ability to customize software to specific agency needs
- Reduced cost and less dependency on imported technology and skills

It applies the sequential approach of the traditional software life-cycle paradigm as well as an evolving approach employed in evolutionary software development, particularly, using the incremental model. A version or prototype is used for better understanding of user requirements and evaluation and testing of users.

The model is also based upon decentralized model and introduces collaborative environment allowing independent peer review. Consequently, the Internet is used as a tool to create a central source of information (mail, web, CVS, forums). Design decisions and programming algorithms can be discussed intensely using this medium.

5.2.2 Open Source Development Lifecycle Phases

5.2.2.1 Initiation

In this phase the needs of the Agency are assessed and it is determined whether a particular need can be fulfilled by the development of a new system. During this phase resources are appraised to check its availability for use in the proposed development.

A Project Proposal is required in this phase and it should include; but not limited to, the following:

- **Project Definition:** identify the client, work to be done, goals of the project, scope and primary function of the software to be developed.
- **Evaluation:** determine the existence of available open source software, suitability to requirements and availability of resources.
- **Planning:** estimate the human effort (person-months), and project duration (in calendar time); schedule the tasks and identify interdependencies.
- **Risk Analysis:** identify and assess risk, mitigation of risks.

This proposal shall be evaluated and results of the evaluation should be quantified in a project evaluation document.

5.2.2.2 Requirements Analysis

In order to have a thorough understanding of the needs of the proposed system, the existing system, whether manual or automated should be fully analyzed. Also called Systems Analysis, this phase requires extensive information gathering and research as well as sifting through documents, reports, and work papers produced by existing systems and extensive interviews with data users and data owners. Deliverables for this phase are a Software Requirements Specification (SRS) document (Annex C) and a Requirements Validation Report.

5.2.2.3 Design

In this phase, the structure and architecture of the proposed system is defined. All data and information gathered in the previous phase shall be used to produce the logical and physical design specifications of the system. Expected outputs of this phase are entity-relationship diagrams, data flow diagrams, program structure charts, system flowcharts, and system infrastructure designs. All these technical models should satisfy the requirements of the proposed system and must be presented thru a design specification document (Annex D).

5.2.2.4 Implementation

During this phase design specifications are translated into the software program. Deliverables of this phase include the initial version of the software, its code, and the coding manual.

5.2.2.5 Testing

This phase requires the testing of the software to make sure that it performs in accordance with both technical and functional business requirements. The system is tested internally by the development team to detect errors and make sure that they are fixed.

After the internal testing, a test version is released for end-users for evaluation. End-users may propose new features or enhancements and/or discover and suggest bug fixes. Required documents for this phase are a software test plan document (Annex E) and a test evaluation document.

5.2.2.6 Release

The system is deployed in its target environment during this stage.

5.2.2.7 Support and Maintenance

In this phase user training and on-going system maintenance is done to ensure the systems' use and functionality. It shall include advanced user training and studies on possible system enhancements. A Users Manual should be provided for all users.

5.2.3 Basic Roles

Executive Committee (EXECOM) Members:

In the development and/or acquisition of new systems, several changes in the operational policies and implementing guidelines need to be applied. EXECOM Members will be responsible for the approval of proposals and changes as well as the allotment of budget.

Management Information System Service (MISS):

MISS is responsible for evaluating the project proposals and recommending what type of project is needed. They shall also formulate the Functional Systems Specification for the system and conduct the initial Requirements Elicitation and Evaluation.

Further, MISS shall be in charge of the Systems Administration and Security once the system is deployed.

Data Owner:

First and foremost, the data owner is responsible for crafting the Project Proposal. They shall also provide all necessary documents and/or reports needed in the Requirements Analysis.

The data owner is also in-charge of the database administration and management of the system. Since the ownership of data and the system is under their supervision, they are the cost center for the Systems Development. There should be a focal person assigned as the **User-level Database Administrator**. It will be his/her responsibility to look for errors and possible system improvements as well as the conduct of new studies and proposals. He is also responsible for the updating of the systems (program syntax). He has the authority to give access levels to the personnel who wishes to access the system.

Another part of the data owners is the **Data User** (Management, Public). They are the ones who make use of the systems data for viewing, analysis and public transparency.

Annex F details all roles.

5.2.4 Software Development Lifecycle Requirement

Each phase in the lifecycle requires:

- Quality Assurance procedures running in parallel with the lifecycle activities
- Verification/Validation report to be conducted by the QA group
- Documentation

5.2.5 Analysis and Design Technique

Open Source projects, as with proprietary projects, require a level of requirements analysis, design and modeling to successfully implement a solution. Unified Modeling Language (UML) is the definitive approach to ensure that the development

incorporates sound processes, robust architecture and reusability. Thus, the Agency shall adopt UML 1.4 by Booch, Rumbaugh and Jacobson as the standard modeling notation.

UML is the standard modeling language set by the Object Management Group (OMG) in 1997 and also the industry standard language for specifying, visualizing, constructing and documenting the artifacts of software systems. It supports iterative and incremental development process and is largely process-independent, meaning that it is not tied to any particular software development life cycle.

UML defines notations and semantics for the following types of problem solution:

- Use Case Model – describes boundary and interaction between users and the system.
- Collaboration Model – describes how objects in the system interact to accomplish tasks.
- Dynamic Model – State charts describe states that classes assume over time. Activity diagram describe the workflow the system will accomplish.
- Logic or Class Model – describes the classes and objects of the system.
- Physical Component Model – describe software, and sometimes hardware, of the system.
- Physical Deployment Model - describes the systems physical architecture and component deployment on system hardware.

5.2.6 Software Licensing and Ownership

The Agency shall have full, uncontestable rights to all information and data generated by the system and to the design and source code of the system.


5.2.7 Development Tools

The Agency shall use solid free programming languages, platform and tools for software development. This allows affordable software for the government and access to software without costly licensing implications.

6. EFFECTIVITY

This circular takes effect immediately.

Issued in Quezon City this 20 of January 2005.


CORAZON JULIANO-SOLIMAN
Secretary
Department of Social Welfare and Development

*Adapted with major revisions from the DOST Software Standard Template

ANNEX A

DEFINITION OF TERMS

Agency – The Department of Social Welfare and Development; or any of its offices or institutions.

Architecture - A design. The term *architecture* can refer to either hardware or software, or to a combination of hardware and software. The architecture of a system always defines its broad outlines, and may define precise mechanisms as well.

An open architecture allows the system to be connected easily to devices and programs made by other manufacturers. Open architectures use off-the-shelf components and conform to approved standards. A system with a *closed architecture*, on the other hand, is one whose design is *proprietary*, making it difficult to connect the system to other systems.

Data - Distinct pieces of information usually formatted in a special way. Data can exist in a variety of forms -- as numbers or text on pieces of paper, as bits and bytes stored in electronic memory, or as facts stored in a person's mind.

Data Owner - Individuals, normally managers or directors, who have responsibility for the integrity, accurate reporting and use of computerized data.

Functional System Specification (FSS) – The Functional System Specification (FSS) is the major requirement in the development or acquisition of individual application systems. The major user, or application owner, is responsible for developing the FSS. The MISS technical staff provides assistance.

A Functional System Specification describes an application in terms of functional requirements and interfaces. It identifies the major entities and lists critical data elements. It describes the inputs, processes and outputs of a system focusing on what is important to achieve the goals of the application system.

The basis of the FSS is the business process analysis. The starting point in any applications is to understand the context of the information, not the organization. The primary focus is to define the "functions" needed to deliver a certain product or service to achieve the goals of the organization.

The FSS is based primarily on data and how data is translated or converted into other forms. While the present organizational structure may be considered during the development of the FSS, the FSS is not limited by it. The FSS may demand a change in the way things are done regardless of the organizational structure. Thus, there is an opportunity for change management when implementing application systems.

Two types of business modeling techniques may be used as part of the FSS:

- Entity Relationship Diagram (ERD). ERD captures the organization's business functions and identify business processes. It depicts the entities perceived as

important to the running of the business, such as people, things or concepts on which information is needed.

- Data Flow Diagram (DFD). A DFD depicts how data is processed. It is based on policies and procedures and shows the processes and outputs.

Life Cycle - Period of time that starts when a software product is conceived and ends when the software is no longer available for use.

Open Source - a term used to describe a tradition of open standards, shared source code, and collaborative software development.

Phase - Period of time during the life cycle of a project in which a related set of software engineering activities is performed. Phases may overlap.

Quality - the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs.

Requirements Elicitation - the identification of specific requirements through a wide variety of techniques.

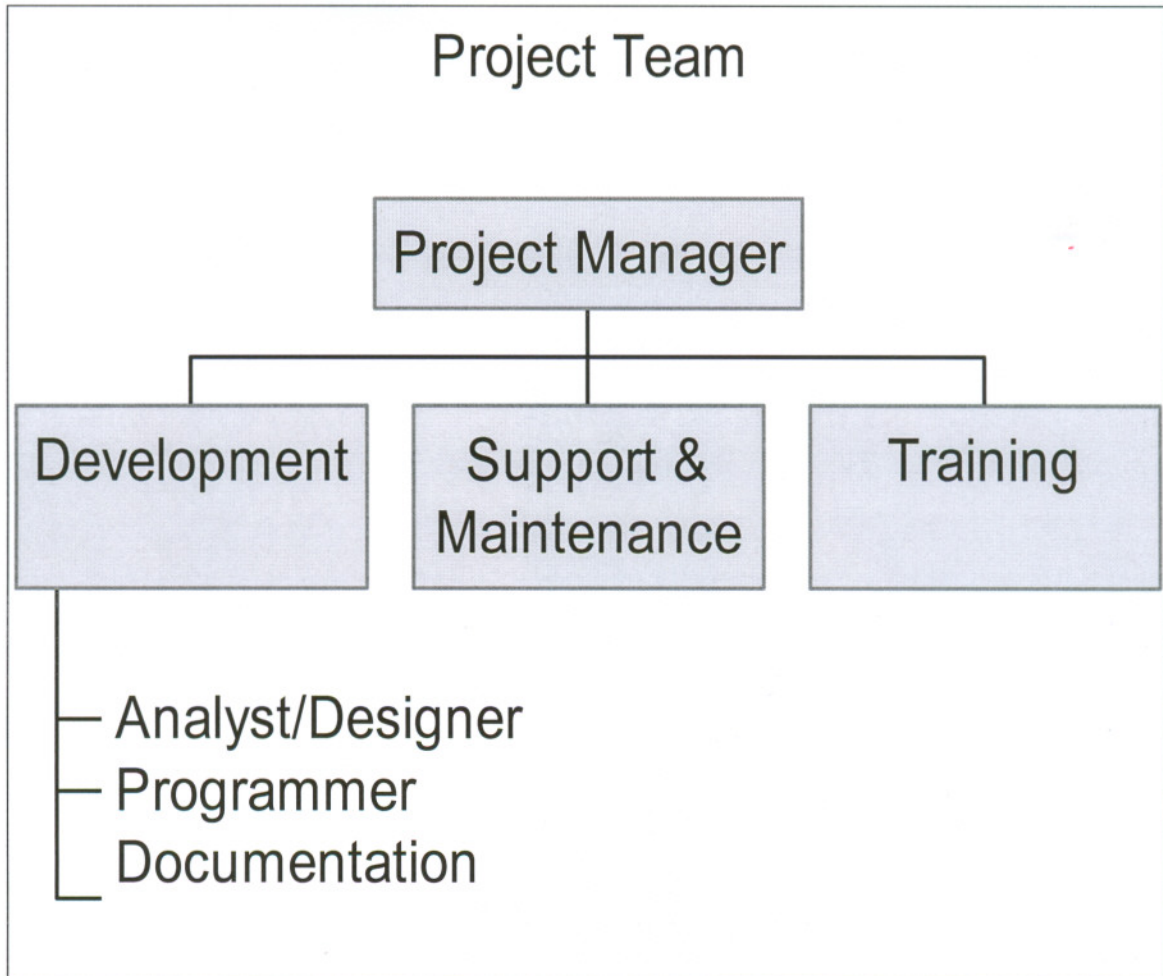
Requirements Evaluation - the process of validating gathered requirements to provide an accurate account of all client requirements.

Software - programs, procedures, rules and any associated documentation pertaining to the operation of a computer system

User - An individual who uses a computer. This includes expert programmers as well as novices. An *end user* is any individual who runs an application program.

ANNEX B

SOFTWARE PROJECT DEVELOPMENT TEAM



ANNEX C

SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT TEMPLATE (To be accomplished by MISS/Outsourced Developer with assistance from the Data Owner)

1. Introduction

1.1 Purpose of the document

Describe the goal and purpose of the project including the intended audience.

1.2 Scope of the software

Describe the domain problem and what part of it will be addressed. Indicate the limitations and disclaimers the reader should know.

1.3 Definitions, acronyms and abbreviations

1.4 References

1.5 Overview of the document

2. Overall description

2.1 Product perspective

Describe the relation to the other components. Describe interfaces to other software, hardware and users.

2.2 Product functions

Describe the main functions the specified product must perform.

2.3 User characteristics

Who will use the system? What training is needed?

2.4 General constraints

Describe constraints that apply and why they exist.

2.5 Assumptions and dependencies

List factors which can affect the requirements. This is not design constraints, but rather things that if they are changed, you have to change the requirements. For instance, you might assume that data is updated each second, that back-up is taken weekly or that a certain version of the browser is available.

3. Specific requirements

3.1 Interface requirements

This section is applicable if more detail than the description in 2.1 is needed.

3.1.1 User interfaces

Describe required user interface. This is not the design of the user interface.

3.1.2 Hardware interfaces

Describe the things you need. Probably only a standard WS with keyboard and mouse.

3.1.3 Software interfaces

Detailed description of input and output items to the system.

3.1.4 Communication interfaces

Describe intranet/internet connection

3.2 Functional requirements (function-oriented)

3.2.1 Information flows

Describe how information is flowing between system parts to realize the function.
Use sequence diagram.

3.2.2 Process description

Describe processes realizing the function in terms of input, output and action.
Use use-case descriptions.

3.2.3 Data construct specification

Describe data in terms of data requirements. Use class diagram

3.3 Performance requirements

Specify both static and dynamic numerical constraints such as: number of users, response time, max. simultaneous users, amount of information to be handled. This is probably most interesting for the optimization group.

3.4 Design constraints

In principle, the SRS should not deal with design, but there might be requirements that are known that will constrain the designers' work, for instance, use of a certain standard. In your case the negotiated protocols might impose constraints.

3.5 Software system attributes

System attributes or quality factors which have requirements, for example:

- ♦ Reliability
- ♦ Availability
- ♦ Security
- ♦ Maintainability
- ♦ Portability
- ♦ Usability

3.6 Other requirements

Things that did not fit above. If this section becomes large, maybe you should consider changing the document structure.

Appendices

ANNEX D

DESIGN SPECIFICATION DOCUMENT TEMPLATE

(To be accomplished by MISS or Outsourced Developer)

1. Introduction

1.1 Purpose of the document

Describe the goal and purpose of the project including the intended audience.

1.2 Scope of the software

Describe the domain problem and what part of it will be addressed. Indicate the limitations and disclaimers the reader should know.

1.3 Definitions, acronyms and abbreviations

1.4 References

1.5 Overview of the document

2. Preliminary Design

2.1 Conceptual High Level Architecture:

Include high level graphical representation depicting the overall application architecture. The design should use package diagram and design class diagram. Dependency of design units should be indicated. A design unit is defined as a process or function, object, class, screen, etc.

3. Detailed Design

3.1 Conceptual Low Level Architecture

For process modeling, this section decomposes and describes sub processes (functions). Use sequence diagram.

3.2 Conceptual Data Model

This section includes a graphical representation and a textual description of the logical database structures. The relationships between tables and entities for the overall application should be clear. The table depictions should be populated with data from the data dictionary. This representation should include relationships between tables and external entities as well. Each textual description should include a definition of the entity/table and their relationships.

3.3 Schema Design Diagram

This section shall include a graphical representation of the physical structure of the database tables and a textual description of those tables. Use design class diagram.

3.4 Graphical User Interface Design

In this section rough out the user interface including any application menus, screens or forms, reports, etc. This may include a graphical depiction of application menus, forms, reports, etc.

3.5 Data Dictionary

This section shall include a list of all data elements included in the physical schema. At a minimum, each data element should contain the following:

- ♦ Data Element Name
- ♦ User Defined Name
- ♦ Definition
- ♦ Data Type
- ♦ Data Format/Length
- ♦ Synonyms
- ♦ User Synonyms
- ♦ Specifications/Algorithms for Derived Data

3.6 Data Validation Procedures, Referential Integrity Rules, Approaches to Enforcing Business Rules

Discuss how data integrity will be maintained within the data base.

4 Interface Design

Identify and describe the interface as follows:

- a. Type of interface (such as real-time data transfer, storage-and-retrieval of data, etc.) to be implemented
- b. Characteristics of individual data elements that the interfacing entity(ies) will provide, store, send, access, receive, etc., such as:
 - i. Names/identifiers
 1. Data Element Name
 2. Non-Technical Name
 3. Definition
 4. Data Type (alphanumeric, integer, etc.)
 5. Data Format/Length
 6. Synonyms
 7. Non-Technical Synonyms
 8. Specifications/Algorithms for Derived Data
 9. Range or enumeration of possible values (such as 0-99)
 - ii. Accuracy (how correct) and precision (number of significant digits)
 - iii. Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the data element may be updated and whether business rules apply
 - iv. Security and privacy constraints
- c. Sources (setting/sending entities) and recipients (using/receiving entities)
- d. Characteristics of data element assemblies (records, messages, files, arrays, displays, reports, etc.) that the interfacing entity(ies) will provide, store, send, access, receive, etc., such as:
 - i. Names/identifiers
 1. Technical Name (e.g., record or data structure name in code or database)
 2. Non-Technical Name Synonyms
 3. 2) Data elements in the assembly and their structure (number, order, grouping)
 4. 3) Medium and structure of data elements/assemblies on the medium

5. 4) Visual and auditory characteristics of displays and other outputs (such as colors, layouts, fonts, icons and other display elements, beeps, lights)
 - ii. Relationships among assemblies, such as sorting/access characteristics
 - iii. Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the assembly may be updated and whether business rules apply
 - iv. Security and privacy constraints
- e. Sources (setting/sending entities) and recipients (using/receiving entities)
- f. Characteristics of communication methods that the interfacing entity(ies) will use for the interface, such as:
 - i. Communication links/bands/frequencies/media and their characteristics
 1. Message formatting
 2. Flow control (such as sequence numbering and buffer allocation)
 3. Data transfer rate, whether periodic/aperiodic, and interval between transfers
 4. Routing, addressing, and naming conventions
 5. Transmission services, including priority and grade
 6. Safety/security/privacy considerations, such as encryption, user authentication, compartmentalization, and auditing
 - ii. Characteristics of protocols that the interfacing entity(ies) will use for the interface, such as:
 1. Priority/layer of the protocol
 2. Packeting, including fragmentation and reassembly, routing, and addressing
 3. Legality checks, error control, and recovery procedures
 4. Synchronization, including connection establishment, maintenance, termination
 5. Status, identification, and any other reporting features
- g. Other characteristics, such as physical compatibility of the interfacing entity(ies) (dimensions, tolerances, loads, voltages, plug compatibility, etc.)]

Appendices

ANNEX E

SOFTWARE TEST PLAN DOCUMENT TEMPLATE

(To be accomplished by MISS with assistance from the Data Owner)

Introduction

The *Introduction* section of the Software Test Plan (STP) provides an overview of the project and the product test strategy, a list of testing deliverables, the plan for development and evolution of the STP, reference material, and agency definitions and acronyms used in the STP.

1.1 Objectives

(Describe, at a high level, the scope, approach, resources, and schedule of the testing activities. Provide a concise summary of the test plan objectives, the products to be delivered, major work activities, major work products, major milestones, required resources, and master high-level schedules, budget, and effort requirements.)

1.2 Testing Strategy

Testing is the process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item.

(This may appear as a specific document (such as a Test Specification), or it may be part of the organization's standard test approach. For each level of testing, there should be a test plan and an appropriate set of deliverables. The test strategy should be clearly defined and the Software Test Plan acts as the high-level test plan. Specific testing activities will have their own test plan. Refer to section 5 of this document for a detailed list of specific test plans.)

Specific test plan components include:

- ♦ *Purpose for this level of test,*
- ♦ *Items to be tested,*
- ♦ *Features to be tested,*
- ♦ *Features not to be tested,*
- ♦ *Management and technical approach,*
- ♦ *Pass / Fail criteria,*
- ♦ *Individual roles and responsibilities,*
- ♦ *Milestones,*
- ♦ *Schedules, and*
- ♦ *Risk assumptions and constraints.*

1.3 Scope

(Specify the plans for producing both scheduled and unscheduled updates to the Software Test Plan (change management). Methods for distribution of updates shall be specified along with version control and configuration management requirements must be defined.)

1.4 Reference Material

(Provide a complete list of all documents and other sources referenced in the Software Test Plan. Reference to the following documents (when they exist) is required for the high-level test plan:

- ♦ *Project authorization,*
- ♦ *Project plan,*
- ♦ *Quality assurance plan,*

- ♦ Configuration management plan,
- ♦ Organization policies and procedures, and
- ♦ Relevant standards.)

1.5 Definitions and Acronyms

(Specify definitions of all terms and agency acronyms required to properly interpret the Software Test Plan. Reference may be made to the Glossary of Terms on the IRMC web page.)

TEST ITEMS

(Specify the test items included in the plan. Supply references to the following item documentation:

- ♦ Requirements specification,
- ♦ Design specification,
- ♦ Users guide,
- ♦ Operations guide,
- ♦ Installation guide,
- ♦ Features (availability, response time),
- ♦ Defect removal procedures, and
- ♦ Verification and validation plans.)

2.1 Program Modules

(Outline testing to be performed by the developer for each module being built.)

2.2 User Procedures

(Describe the testing to be performed on all user documentation to ensure that it is correct, complete, and comprehensive.)

2.3 Operator Procedures

(Describe the testing procedures to ensure that the application can be run and supported in a production environment (include Help Desk procedures)).

3. Features To Be Tested

(Identify all software features and combinations of software features to be tested. Identify the test design specifications associated with each feature and each combination of features.)

4. FEATURES NOT TO BE TESTED

(Identify all features and specific combinations of features that will not be tested along with the reasons.)

5. APPROACH

(Describe the overall approaches to testing. The approach should be described in sufficient detail to permit identification of the major testing tasks and estimation of the time required to do each task. Identify the types of testing to be performed along with the methods and criteria to be used in performing test activities. Describe the specific methods and procedures for each type of testing. Define the detailed criteria for evaluating the test results.)

(For each level of testing there should be a test plan and the appropriate set of deliverables. Identify the inputs required for each type of test. Specify the source of the input. Also, identify the outputs from each type of testing and specify the purpose and format for each test output.

Specify the minimum degree of comprehensiveness desired. Identify the techniques that will be used to judge the comprehensiveness of the testing effort. Specify any additional completion criteria (e.g., error frequency). The techniques to be used to trace requirements should also be specified.)

5.1 Component Testing

(Testing conducted to verify the implementation of the design for one software element (e.g., unit, module) or a collection of software elements. Sometimes called unit testing. The purpose of component testing is to ensure that the program logic is complete and correct and ensuring that the component works as designed.)

5.2 Integration Testing

(Testing conducted in which software elements, hardware elements, or both are combined and tested until the entire system has been integrated. The purpose of integration testing is to ensure that design objectives are met and ensures that the software, as a complete entity, complies with operational requirements. Integration testing is also called System Testing.)

5.3 Conversion Testing

(Testing to ensure that all data elements and historical data is converted from an old system format to the new system format.)

5.4 Job Stream Testing

(Testing to ensure that the application operates in the production environment.)

5.5 Interface Testing

(Testing done to ensure that the application operates efficiently and effectively outside the application boundary with all interface systems.)

5.6 Security Testing

(Testing done to ensure that the application systems control and auditability features of the application are functional.)

5.7 Recovery Testing

(Testing done to ensure that application restart and backup and recovery facilities operate as designed.)

5.8 Performance Testing

(Testing done to ensure that that the application performs to customer expectations (response time, availability, portability, and scalability)).

5.9 Regression Testing

(Testing done to ensure that that applied changes to the application have not adversely affected previously tested functionality.)

5.10 Acceptance Testing

(Testing conducted to determine whether or not a system satisfies the acceptance criteria and to enable the customer to determine whether or not to accept the system. Acceptance testing ensures that customer requirements' objectives are met and that all components are correctly included in a customer package.)

5.11 Beta Testing

(Testing, done by the customer, using a pre-release version of the product to verify and validate that the system meets business functional requirements. The purpose of beta testing is to detect application faults, failures, and defects.)

6. PASS / FAIL CRITERIA

(Specify the criteria to be used to determine whether each item has passed or failed testing.)

6.1 Suspension Criteria

(Specify the criteria used to suspend all or a portion of the testing activity on test items associated with the plan.)

6.2 Resumption Criteria

(Specify the conditions that need to be met to resume testing activities after suspension. Specify the test items that must be repeated when testing is resumed.)

6.3 Approval Criteria

(Specify the conditions that need to be met to approve test results. Define the formal testing approval process.)

7. Testing Process

(Identify the methods and criteria used in performing test activities. Define the specific methods and procedures for each type of test. Define the detailed criteria for evaluating test results.)

7.1 Test Deliverables

(Identify the deliverable documents from the test process. Test input and output data should be identified as deliverables. Testing report logs, test incident reports, test summary reports, and metrics' reports must be considered testing deliverables.)

7.2 Testing Tasks

(Identify the set of tasks necessary to prepare for and perform testing activities. Identify all intertask dependencies and any specific skills required.)

7.3 Responsibilities

(Identify the groups responsible for managing, designing, preparing, executing, witnessing, checking, and resolving test activities. These groups may include the developers, testers, operations staff, technical support staff, data administration staff, and the user staff.)

7.4 Resources

(Identify the resources allocated for the performance of testing tasks. Identify the organizational elements or individuals responsible for performing testing activities. Assign specific responsibilities. Specify resources by category. If automated tools are to be used in testing, specify the source of the tools, availability, and the usage requirements.)

7.5 Schedule

(Identify the high level schedule for each testing task. Establish specific milestones for initiating and completing each type of test activity, for the development of a comprehensive plan, for the receipt of each test input, and for the delivery of test output. Estimate the time required to do each test activity.)

(When planning and scheduling testing activities, it must be recognized that the testing process is iterative based on the testing task dependencies.)

8. Environmental Requirements

(Specify both the necessary and desired properties of the test environment including the physical characteristics, communications, mode of usage, and testing supplies. Also provide the levels of security required to perform test activities. Identify special test tools needed and other testing needs (space, machine time, stationary supplies. Identify the source of all needs that is not currently available to the test group.)

8.1 Hardware

(Identify the computer hardware and network requirements needed to complete test activities.)

8.2 Software

(Identify the software requirements needed to complete testing activities.)

8.3 Security

(Identify the testing environment security and asset protection requirements.)

8.4 Tools

(Identify the special software tools, techniques, and methodologies employed in the testing efforts. The purpose and use of each tool shall be described. Plans for the acquisition, training, support, and qualification for each tool or technique.)

8.5 Publications

(Identify the documents and publications that are required to support testing activities.)

8.6 Risks and Assumptions

(Identify significant constraints on testing such as test item availability, test resource availability, and time constraints. Identify the risks and assumptions associated with testing tasks including schedule, resources, approach and documentation. Specify a contingency plan for each risk factor.)

9. Change Management Procedures

(Identify the software test plan change management process. Define the change initiation, change review, and change authorization process.)