

TABLE OF CONTENTS

INTRODUCTION	1
Concept of Phased Approach	2
Key Features of the RSDM	2
Description of the Phases	3
Phase 1. Identification of User Needs	5
Identification of User Needs Activities	5
Describe Current Environment	5
Document User Needs	5
Define Terminology and Elements	5
Compile Documents	6
Accept User Statement of Needs	6
Phase 2. Requirements Definition	7
Requirements Definition Activities	7
Develop Preliminary Statement of Work	7
Review Statement of Work	7
Develop Model of Current User Processes	8
Develop Data Relationship Model	8
Define Domain of Change	8
Develop Logical Model of Future Environment	8
Define Requirements	9
Review Requirements	9
Phase 3. Database Design	10
Database Design Activities	10
Design Initial Database	10
Perform Optimization	10
Define Conversion Requirements	10
Define Archiving Requirements	11
Define Database Security Requirements	11
Review Database	11
Phase 4. Detailed Design	12
Detailed Design Activities	12
Expand Project Plan	12
Develop Programmer's Handbook	12
Design Common Standards/Templates	13
Develop Component Design Hierarchy	13
Write Function Specifications/Program Specifications	13
Phase 5. Software Construction	14
Software Construction Activities	14
Expand Software Construction Plan	14
Establish Environment Libraries	14
Present Overview to Programming Team	15
Perform Construction of Components	15
Perform Final System Test	15
Complete System Documentation and User Manuals	15
Complete Certification Test Cases	16
Develop Training Materials	16
Phase 6. Software Certification	17
Software Certification Activities	17
Perform Alpha Certification Testing	17
Conduct Alpha Test Review and Respond Accordingly	17
Perform Beta Certification Testing	18
Conduct Beta Test Review and Respond Accordingly	18
Phase 7. Installation	19
Installation Activities	19

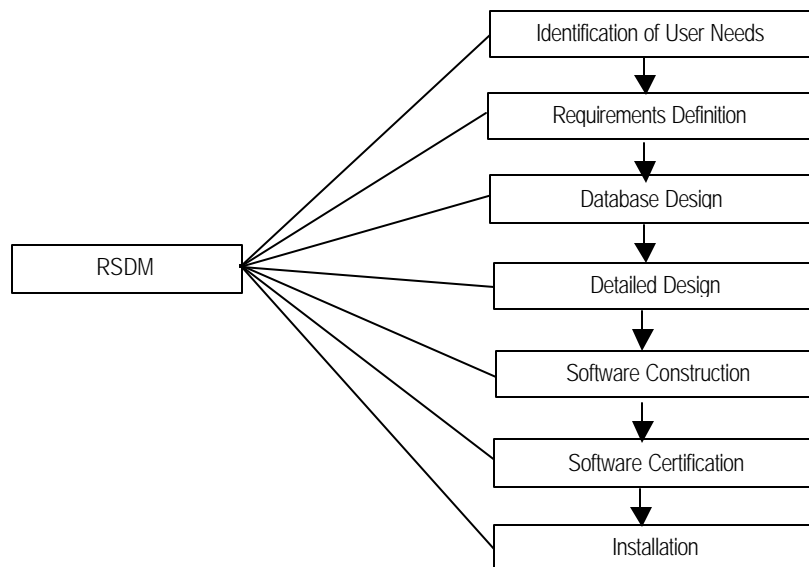
RADIX Systems Development Methodology (RSDM)
by RADIX Systems Services Corporation

Finalize Installation / Conversion / Training Plan	19
Conduct Management and Operations Training Programs	19
Perform System Conversion	19
Install System	20
Certify System.....	20

INTRODUCTION

All development projects must have defined limits as distinct from being on-going or continuing. RADIX, being a contractor, has all the more reasons to think in terms of fixed beginning and ending because of pre-defined contract deliverables and expectations. However, the concept of boundaries should not limit the functionality of a system from between-project revisions or future enhancements. These opposing views are the main reasons why system development projects must follow a phased approach.

RSDM (RADIX Systems Development Methodology) is intended as a guide for all personnel involved in the different phases of systems development and includes techniques to carry out the approach. As a guide, it provides recommendations and not limitations. Development projects vary in scope, size, complexity and many other aspects. Therefore, each project must be assessed to determine how RSDM is to be applied to the different phases of the project. The figure below summarizes the major RSDM project phases.



This manual describes the phased approach and provides guidelines within which distinct project milestones can be defined. It uses technical terms and presentation documents familiar to development persons. However, when the output of the phase is to be presented to the users, the output and terms used should be translated into formats and terms easily understood by the end users. A good understanding of the requirements and expectations can only be achieved through confirmation from the users. Hence, we need to be understood and using jargons and objects not familiar to the users will only push us away from our objectives. By simplifying the output to the level of the users, the confirmation and approval processes required by the development phases will be expedited.

Concept of Phased Approach

Generally, it is a practice to define project activities according to two basic criteria:

- Projects are subdivided at points where the nature of the work changes. Usually this involves a shift in the types of activity and skills required.
- Projects are also subdivided according to logical points for management review, approval and resource management.

Many of the software development activities are cyclical or repetitive in nature. Because of this nature, development projects are better controlled using a phased approach. A phased approach implies the movement of activities from one area of emphasis to the next. It differs from hierarchical approach wherein succeeding activities are not started until the previous activity has finished. Although we are presenting a model depicting an orderly progression of a project with specific periods of time, it is, in fact, interactive and overlapping. By phased approach, we can fine-tune the model according to the project need in terms of availability of resources, both from the user side and that of RADIX.

As long as dependencies and control points are defined, any development project can be planned in many ways beneficial to both parties. The progression from one phase to the other depends on checkpoint information available. The information establishes the baseline for that phase and from which the next phase can proceed. Problems, which may impact the quality and efficiency of subsequent phases, are thus avoided. As work progresses and status is monitored, alternative solutions can be planned and critical decisions can be made, minimizing risks, in terms of time and costs.

Key Features of the RSDM

RSDM phased-approach forces planners to identify activities to the level of skills and expected due dates. Each task has defined inputs and outputs. With the guide, the team is able to dedicate their time to the specific requirements of the system or project. This approach has the following features:

- **Consistency** - Everyone involved in the project, including the users speaks the same language. Consistent symbols, terminology and methods enable the development of measurable standards and a framework that paves the way to repeatable results.
- **Quality** - With tested standards and templates, we are assured of high quality output. The recurring reviews not only provide a means of early problem detection but also provide a discipline of good documentation along the way.
- **Realistic Estimates** - There are a lot of variables involved in project planning and estimating. Variables which may change as the project progresses. In RSDM, planning and estimating must be done in all levels. General estimates for the total project and detailed estimates before each phase is started.
- **User Involvement** - RSDM is designed to facilitate the communication between developer and the users. The most knowledgeable person and ultimate approver of the system are the users. When users are involved, the probability of failure is less because the users will be able to tell us what they want, check on our understanding and expectations are constantly redefined.

-
- **Implementation and Training** - Activities within this approach involves a lot of meetings with the users. As the design is being done and presented to the users from time to time, the users are gaining an insight on how the final system will look. This way, we are preparing them earlier without the burden or pressure the usual “install-train-implement” activities put on both the developer and user.
 - **Baseline establishment** - Each phase in the process is marked by clearly identified and agreed upon point of reference called the baseline. Each baseline is from where the next phase is built. By establishing baselines, we have specific points where changes can be formally evaluated, implemented and monitored.
 - **Reviews** - With phase reviews, we are able to establish baselines and assess the status of the project. Communication cannot be overemphasized. Reviews are important part in providing project visibility for proper project control and decision-making.
 - **Change Control** - Even with defined-requirements and approved baselines, changes will always be there. A good change control assures us of a smooth implementation of changes. Changes must be evaluated for their impact on the system, schedules and cost.

Description of the Phases

A brief description of each phase follows:

- **Identification of User Needs** - Any systems development project is a result of a user need. When and who documents the need differ from one project to the other. In most cases, user needs have to be defined before any development contract is entered into by RADIX. However, this phase has to be performed again, to revalidate expectations.

It is the user who presents the need. Definition of current functional activities will establish the baseline for future activities.

- **Requirements Definition** - Heavy interaction with the user is necessary to establish a set of specified requirements to which both parties can agree on the basis of specific acceptance criteria.

From these requirements, solutions are identified and evaluated. Risks and benefits of each solution are determined and schedules are redefined.

After the approach is defined, the requirements are detailed. This phase will ensure that requirements are defined according to the expectations of the users. Although there have been functions previously defined, the end result of this phase is to put into screen and report layouts, the detail of the system.

Although part of the requirements may involve the acquisition of new hardware/software or hiring of additional personnel,

planning for such additional resources, although material to the total project, has been excluded from this manual.

- **Database Design**

- A good database design can very well dictate the success of the project. In addition to this, it facilitates future modifications. However, in most cases, the users do not understand the database conventions developers use. In some cases, the users do not want to be involved in the details. Because of its importance, presentations must be adjusted to the level the users will most be comfortable with.

In other methodologies, this phase is part of the Detailed Design. In RSDM, Database Design is separated because database is a different view by itself. Designing the process details of the system at the same time designing the database may result in a process-driven database design.

- **Detailed Design**

- This phase involves the conversion of the requirements to a system design, which can be broken down into programmable components.

This is also where common procedures and templates are defined.

- **Software Construction**

- The detailed design is transcribed into source codes, assembled and tested.

Before any construction is initiated, it is important that a plan is made to reflect strategies that would increase productivity and quality within pre-agreed schedules.

- **Software Certification**

- A two-phased approach (Alpha and Beta) is used to ensure that the designed system correctly satisfies the requirements. A certification team is responsible for developing the test plans and for conducting or overseeing the required tests.

- **Installation**

- Upon certification, the system is ready for installation. Installation includes user training, conversion of existing databases and activities need to prepare the system for production.

Phase 1. Identification of User Needs

This phase starts the software development life cycle. The user defines the current functional tasks, a glossary of terms and a statement of needs. These three items will help the user explain what is needed.

For RADIX, this phase may be considered as a pre-project phase essential for the preparation of proposals and contracts. Most customers would prefer to see the overall schedules and cost before signing a development contract. During proposal preparation, the depth by which needs are established may be not as detailed as that done when the project starts.

When the project starts, this phase is done, in a more detailed manner, to make sure that both parties are aware of the scope of the contract. Although it is the user who is responsible at the start of this phase, the compilation of the output is on the developer.

For this manual, discussions will tackle the activities done for this phase within the project duration.

1.0 Identification of User Needs Activities		
1.1	<p>Describe Current Environment</p> <p>Responsible – User</p>	<p>Description/Purpose This task includes the steps necessary to document the user's current tasks. The user describes the work done by the organization. All relevant tasks and procedures needed to accomplish their work are documented.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> Describe User Organization. Document Existing Mechanized Support. Describe User Tasks. 	<p>Input: Present procedures and operational environment</p> <p>Output: Organizational Structure Description of Current Operation</p>
1.2	<p>Document User Needs</p> <p>Responsible – User</p>	<p>Description/Purpose The user will compile the reasons for changing the current operating environment. Also, the user will document the expected impact of the desired changes.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> State User Assumptions. List any assumptions that were made during the process of defining user's needs. List User Needs. Areas where a user may have needs include new data, new processes, interfaces, and work performances. Describe Constraints. List any constraints on potential solutions for the user's needs. List User's Acceptance Criteria. Describe the User's View of the Future Operation/Environment. 	<p>Input: User Needs User's future operational systems plan</p> <p>Output: Assumptions User expectations Problem/opportunity statement</p>
1.3	<p>Define Terminology and Elements</p> <p>Responsible – User</p>	<p>Description/Purpose The terminology and data elements used by the user must be defined to assure a common understanding with the developer.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> Compile User Terminology Identify Data Elements. Include data element characteristics (definition, size, data type, source, volume, custodian, and dependencies). 	<p>Input: User Definitions User Terminology</p> <p>Output: Data Identification System User Glossary Preliminary data dictionary</p>

1.0 Identification of User Needs Activities		
1.4	<p>Compile Documents</p> <p>Responsible – Developer</p>	<p>Description/Purpose Information gathered in previous tasks is compiled into a single document. This resultant document represents the user's view and it will be used as input in the Requirements Definition phase.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> Tailor User Statement Format. Compile User Needs. 	<p>Input: Description of User Needs User Glossary Problem/Opportunity Statement</p> <p>Output: User Statement of Needs and Environment</p>
1.5	<p>Accept User Statement of Needs</p> <p>Responsible – User/Developer</p>	<p>Description/Purpose This task serves as the phase milestone for the identification of user needs phase. The User Statement of Needs and Environment prepared by the developer is presented to the user management for corrections and subsequent approval.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> Review by Client. Approve Statement of User Needs and Environment 	<p>Input: User Statement of Needs and Environment</p> <p>Output: Accepted User Statement of Needs and Environment</p>

Phase 2. Requirements Definition

The end objective of this phase is to come up with a Requirements Definition Manual, containing the External Design of the system. From the User Statement of Needs, the current user environment is analyzed together with the needs to come up with proposed new user environment. After the general approach is defined, the requirements are detailed. This phase will ensure that requirements are defined according to the expectations of the users. Although there have been functions previously defined, the end result of this phase is to put into screen and report layouts, the detail of the system.

In other methodologies, detailing to screen and report level is part of a phase they call Detailed Design. However, statistics, based on past RADIX development projects, prove that unless the users see the actual running screens and reports, a lot of changes and misinterpretations are encountered along the system's life cycle. Although changes can never be avoided, presenting the requirements in a prototype form up-front will lessen the need for changes, provide a better basis for plans, schedules, cost analysis. Also, if this phase was able to capture all requirements of the system into the Requirements Definition Manual, the manual, with minor revisions, should be enough to present itself as the system's User Manual when system is finally documented.

Since the output of this phase will be the most important baseline from where succeeding phases will refer to, it is necessary that enough time be spent for iterative analyses and reviews. Consequently, heavy interaction between user and developer is obvious. The user is the key factor to this phase. It is important that the developer makes the user at ease with the activities without imposing on them. Users are threatened when presented with technical I.T. terms. Getting the users to participate in the activities can only be achieved if the developer goes down or up to their level.

2.0 Requirements Definition Activities		
2.1	<p>Develop Preliminary Statement of Work</p> <p>Responsible – Developer</p>	<p>Description/Purpose A Statement of Work (SOW) consists of a list of deliverables, statement of objectives, delineation of responsibilities and estimates. Also, a management plan should be developed containing standards, status reporting and change control procedures. The plans should be scaled to the magnitude and the complexity of the user's needs and operational environment. Eventually, the schedule for the Requirements Definition phase is constructed.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Develop Statement of Work. 2. Establish Standards. 3. Establish Status Reporting Procedures. 4. Establish Change Control Procedures. 5. Construct Requirements Definition Analysis Schedule 	<p>Input: User Statement of Needs and Environment Scope of Analysis</p> <p>Output: Preliminary Statement of Work Management Plan Requirements Definition Analysis Schedule</p>
2.2	<p>Review Statement of Work</p> <p>Responsible – User/Developer</p>	<p>Description/Purpose The Statement of Work, Management Plan and Requirements Definition Schedule are presented both to the developer's management team and the users. Revisions to the plans are discussed and incorporated. Final plans are then, approved by the users.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Review of Plan by the Developer's Management Team. 2. Review of Plan by the Users. 3. Revise Plans. 	<p>Input: Statement of Work Management Plan Requirements Definition Schedule</p> <p>Output: Approved Statement of Work Approved Management Plan Approved Requirements Definition Schedule</p>

2.0 Requirements Definition Activities		
2.3	<p>Develop Model of Current User Processes</p> <p>Responsible – User/Developer</p>	<p>Description/Purpose</p> <p>A logical model of the current user processes is developed. The description of the current user environment serves as a basis for developing the logical model. To assure correctness of understanding, the user reviews and approves the model.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Develop Model of Current Physical Processes. 2. Review Physical Model with Users. 3. Revise Current Physical Model. 4. Develop Logical Model of Current Processes. 5. Review Logical Model with Users. 6. Revise Current Logical Model. 	<p>Input:</p> <p>Accepted User Statement of Needs and Environment Approved Management Plans Iterative user interface</p> <p>Output:</p> <p>Approved Current Logical Model</p>
2.4	<p>Develop Data Relationship Model</p> <p>Responsible – User/Developer</p>	<p>Description/Purpose</p> <p>A model of the relationships among the data elements is made. Data elements are defined in terms of other data elements using the process of decomposition. The process continues until a set of non-decomposable data elements is created. The data elements associated with each user process are grouped together. When the process is complete, the basic data elements utilized by the current user processes have been identified and grouped together.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Decompose Data Elements to their Basic Components. 2. Group Related Data Elements. 3. Define Relationships Among Data Element Groups. 4. Compile a Representation of Relationships. 5. Review the Model with the User. 6. Revise Data Relationship Model. 	<p>Input:</p> <p>Current Logical Model Preliminary Data Dictionary Iterative User Interface</p> <p>Output:</p> <p>Data Relationship Model Expanded Preliminary Data Dictionary</p>
2.5	<p>Define Domain of Change</p> <p>Responsible – User/Developer</p>	<p>Description/Purpose</p> <p>The user and developer jointly agree what portions of the current user processes must be changed to meet the user's needs. This task defines only what needs to be changed not how it is to be changed. The domain of change is shown in the current logical model.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Identify Functions/Processes to be Changed. 2. Verify Changes with Users. 3. Revise List of Changes. 4. Annotate Model to Indicate Areas of Change. 	<p>Input:</p> <p>Accepted User Statement of Needs and Environment Current Logical Model Iterative user interface</p> <p>Output:</p> <p>Domain of Change</p>
2.6	<p>Develop Logical Model of Future Environment</p> <p>Responsible – User/Developer</p>	<p>Description/Purpose</p> <p>The user and developer jointly agree what portions of the current user processes must be changed to meet the user's needs. This task defines only what needs to be changed not how it is to be changed. The domain of change is shown in the current logical model.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Develop New Data Flows within the Domain of Change. 2. Develop New Logical Model. 3. Revise Data Relationship Model. 	<p>Input:</p> <p>Domain of Change Data Relationship Model</p> <p>Output:</p> <p>New Logical Model</p>

2.0	Requirements Definition Activities	
2.7	<p>Define Requirements</p> <p>Responsible – User/Developer</p>	<p>Description/Purpose</p> <p>The user and developer jointly develop a set of requirements, which will bring about the changes depicted in the New Logical Model. Each individual requirement is jointly signed and has its own acceptance criteria.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Write Individual Specifications (screens, reports, validations) 2. Validate Specifications 3. Obtain User Concurrence on Individual Specifications 4. Compile Requirements Definition Manual. 	<p>Input:</p> <p>New Logical Model Data Relationship Model Expanded Data Dictionary</p> <p>Output:</p> <p>Requirements Definition Manual</p>
2.8	<p>Review Requirements</p> <p>Responsible – User/Developer</p>	<p>Description/Purpose</p> <p>The task serves as the phase review for the Requirements Definition phase. The Requirements Definition Manual is submitted to the users for review and approval.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Review by Developer Management. 2. Review by Users. 3. Revise Requirements Definition Manual 4. Approve Requirements Definition Manual 	<p>Input:</p> <p>Requirements Definition Manual</p> <p>Output:</p> <p>Approved Requirements Definition Manual</p>

Phase 3. Database Design

As we have indicated before, a good database design is very important, not only to the current systems development project but also, to the future of the said system. The initial design and implementation of a system covers only a portion in its entire life cycle. Future development may involve creation of interfaces to the system and modifications to it. It is important then that the database design can accommodate changes without too much impact to the existing functions. This feature can only be achieved by first, concentrating on data relations before processing requirements are considered.

In the simplest case, database requirements may only require interfaces to current databases. More complex cases require a design of an integrated databases supported by sophisticated data management systems. There are even projects that produce only an integrated database and nothing else. This only proves the importance of the database in any systems development project.

The Requirements Definition phase produces the Data Relationship Model and an Expanded Data Dictionary. This phase uses them as input for the design of the physical databases.

3.0	Database Design Activities	
3.1	Design Initial Database Responsible – Developer	Description/Purpose The developer, using the Data Relationship Model and Expanded Data Dictionary, designs the initial database.
	Subtasks: 1. Group Related Data Fields Together. 2. Categorize Data Fields. 3. Develop Record Layouts. 4. Examine Records Layouts for Redundancy.	Input: Data Relationship Model Expanded Data Dictionary Output: Initial Database Design
3.2	Perform Optimization Responsible – Developer	Description/Purpose Based on the processing needs, the initial database is checked and revised for optimization requirements.
	Subtasks: 1. Establish Priorities for Optimization. 2. Perform File Structure Optimization. 3. Specify Access Keys.	Input: Initial Database Design Requirements Definition Manual Processing Needs of Transactions, Screens, Reports Output: Optimized Database Design
3.3	Define Conversion Requirements Responsible – User/Developer	Description/Purpose Based on the Optimized Database, conversion or set-up requirements, whether from existing databases or hardcopy documents are defined.
	Subtasks: 1. Define Conversion and Set-up requirements.	Input: Optimized Database Existing Database Iterative User Interface Output: Conversion Requirements

3.0 Database Design Activities		
3.4	<p>Define Archiving Requirements</p> <p>Responsible – User/Developer</p>	<p>Description/Purpose The task aims to establish when portions of the database are to be archived.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Establish disk space requirements. 2. Define Archiving Requirements. 	<p>Input: Optimized Database Iterative User Interface</p> <p>Output: Archiving Requirements</p>
3.5	<p>Define Database Security Requirements</p> <p>Responsible – User/Developer</p>	<p>Description/Purpose The task aims to establish the security requirements of the database. Although system functions may already secure the database, there are databases, which can be accessed using interactive data management facilities. It is therefore important that this factor is also considered.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Define Database Security Needs. 2. Define Available Database Security Functions of the DB software or Operating System. 3. Establish Database Security Requirements. 	<p>Input: Optimized Database Requirements Definition Manual (Data Relationship Model – Ownership) Iterative User Interface</p> <p>Output: Database Security Requirements</p>
3.6	<p>Review Database</p> <p>Responsible – User/Developer</p>	<p>Description/Purpose The user must finally agree on the database design approach.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Review Database. 2. Revise Database. 	<p>Input: Optimized Database Database Conversion Requirements Database Archiving Requirements Database Security Requirements</p> <p>Output: Approved Optimized Database Approved Database Conversion Requirements Approved Database Archiving Requirements Approved Database Security Requirements</p>

Phase 4. Detailed Design

This phase marks the transition from the logical definition of the project to the physical definition of the project. Except for the database design, the previous phases' efforts and decisions were aimed at determining a feasible solution to the needs. Why then did we not include database design in this phase? As we have indicated earlier, the reason lies mainly on the need to give the database design a separate focus and importance.

The detailed design phase is predominantly technical, with emphasis on internal designing. User involvement may vary from one project to another. If user has a technical team, the technical team may be involved for purposes of the system's future maintenance requirements. The technical team may contribute in setting common standards, templates and routines.

The design process basically is composed of three major tasks, namely; planning, preparation and detailing. The planning task aims to provide a general view and schedule of the rest of the phases. The results of the Requirements Definition and Database Design phases are reviewed. Technical processes required in the achievement of goals are planned across the remaining phases of the project. The preparation task aims to facilitate the processes involved in the succeeding phases. Basic goal of this task is to come up with standards and processes that will increase the productivity rate in each phase. An activity in one phase may increase hours required but may benefit other activities and phases. The detailing task is the actual writing of specifications for each component of the system.

4.0	<i>Detailed Design Activities</i>	
4.1	<p>Expand Project Plan</p> <p style="text-align: center;">Responsible – User/Developer</p>	<p>Description/Purpose After the external design and database requirements of the system has been designed and concurred with by the users, the planning for the remainder of the project can be planned. Plans include detailed tasks and estimates.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Review Requirements Definition and Database Design. 2. Define Detailed Design Plan. 3. Define Software Construction Plan. 4. Define Testing and Certification Plan. 5. Define Training Plan. 6. Define Conversion Plan. 7. Define Installation Plan. 8. Define Implementation Plan. 9. Define Overall Project Plan. 10. Review Overall Project Plan. 11. Revise Overall Project Plan. 	<p>Input: Requirements Definition Manual Approved Optimized Database Approved Database Conversion Requirements Approved Database Archiving Requirements Approved Database Security Requirements</p> <p>Output: Project Plan Plan Details (Detailed Design, Software Construction, Testing and Certification, Training, Conversion, Installation, Implementation)</p>
4.2	<p>Develop Programmer's Handbook</p> <p style="text-align: center;">Responsible – Developer</p>	<p>Description/Purpose The programmer's handbook is a summary reference for the project team. With so many plans and materials available, the book aims to consolidate everything in one place where team members may refer to when the need arises.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Write a Statement of the Project Purpose. 2. Summarize the Project History. 3. Collect the Project Standards. 4. Define Project Environment. 5. Define Project Procedures. 6. Compile the Programmer's Handbook. 	<p>Input: Project Plan Plan Details (Detailed Design, Software Construction, Testing and Certification, Training, Conversion, Installation, Implementation)</p> <p>Output: Programmer's Handbook</p>

4.0 Detailed Design Activities		
4.3	Design Common Standards/Templates	Description/Purpose This task aims to define in detail, the common standards and templates needed for the project. This usually includes items, which will raise the productivity rates for the succeeding tasks and phases.
	Responsible – Developer	
	Subtasks: 1. Review Plans. 2. Expand Standards/Template Requirements. 3. Collect from Existing Standards/Templates. 4. Design Project Standards/Templates.	Input: Project Plan Plan Details (Detailed Design, Software Construction, Testing and Certification, Training, Conversion, Installation, Implementation) Output: Detailed Design of Standards and Templates.
4.4	Develop Component Design Hierarchy	Description/Purpose The RSDM design strategy is top-down dominated. It involves decomposing design decisions to elementary levels. The concept of decomposition may entail designers to go back to previously decomposed elements for further decomposition or restructuring. Conversion, security and archiving requirements are also part of the system, hence, these requirements should also be considered in this task.
	Responsible – Developer	
	Subtasks: 1. Identify the Sub-processes within the System. 2. Identify the Functions of each Sub-process. 3. Identify Common Functions. 4. Identify Independent Functions. 5. Identify lowest level components.	Input: Requirements Definition Manual Approved Optimized Database Project Plan Plan Details (Detailed Design, Software Construction, Testing and Certification, Training, Conversion, Installation, Implementation) Approved Database Conversion Requirements Approved Database Archiving Requirements Approved Database Security Requirements Output: List of Functions List of Programs/Function List of Sub-routines
4.5	Write Function Specifications/Program Specifications	Description/Purpose This task involves the writing of function and program specifications.
	Responsible –Developer	
	Subtasks: 1. Write Function Specifications. 2. Write Program Specifications.	Input: Requirements Definition Manual Approved Optimized Database Database Conversion Requirements Database Archiving Requirements Database Security Requirements List of Functions List of Programs/Function List of Sub-routines Output: Function Specifications Program Specifications

Phase 5. Software Construction

In this phase, the completed design is transformed into tested computer programs ready for certification. As in other phases, this phase is started with a planning task. The coding and testing of system software must be organized and planned to assure that module construction proceeds in an orderly and efficient manner, makes efficient use of available resources, and complies with scheduled delivery commitments.

The Software Construction Plan has previously been developed in the Detailed Design phase. However, after the Detailed Design phase, a more realistic count has been achieved. Expanding the Software Construction plan will contribute to project visibility and aid in estimating reasonable completion dates based on known complexities and productivity strategies. The plan should address; order of priority, schedule, productivity strategies, skill resources, checkpoints, revisions and machine resources.

A practical starting point in planning is to identify the minimum program configuration. Once the core has been scheduled, other components are prioritized accordingly. Resources required, in the construction of each component, is then determined. Schedules and resources are then balanced to come up with an achievable software construction plan. Also, the plans for the succeeding phases are detailed.

Database construction is part of this phase. The task is usually done concurrent to the construction of standards and templates. While the software construction is being performed, other tasks are likewise being performed such as the preparation of user manual, training materials and test cases.

5.0 Software Construction Activities		
5.1	<p>Expand Software Construction Plan</p> <p>Responsible – User/Developer</p>	<p>Description/Purpose</p> <p>The details coming from the Detailed Design phase will enable the developer to plan, estimate and strategize the approach for the Software Construction phase. Resulting revisions to the Software Construction plan will affect the plans and schedules for the remaining phases. Other plans are also changed.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Define Minimum Program Configuration. 2. Prioritize Programs. 3. Define Resource Requirements. 4. Expand Software Construction Plan. 5. Review Software Construction Plan. 6. Revise Software Construction Plan. 7. Revise Remaining Plans and Schedules. 8. Review Remaining Plans and Schedules. 9. Revise Remaining Plans and Schedules. 	<p>Input:</p> <p>Software Construction Plan List of Programs Available Resources</p> <p>Output:</p> <p>Approved Software Construction Plan Approved Revised Plans (Testing and Certification, Installation, Training, Implementation)</p>
5.2	<p>Establish Environment Libraries</p> <p>Responsible – Developer</p>	<p>Description/Purpose</p> <p>This task will prepare the environment of the project, both on the development environment and documentation environment.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Set up Common Development/Production Libraries. 2. Set up Development Libraries. 3. Set up Production Libraries. 4. Set up Documentation Libraries. 	<p>Input:</p> <p>Software Construction Plan</p> <p>Output:</p> <p>Development/Production/Documentation Libraries</p>

5.0 Software Construction Activities		
5.3	Present Overview to Programming Team Responsible – Developer	Description/Purpose This task aims to familiarize the programming team with the project requirements, plans and strategies. Although they have a programmer's handbook, an initial presentation is very important. There are issues not in the handbook, which the team must be aware of such as schedules and commitments.
	Subtasks: 1. Compile Presentation Materials. 2. Present Project Requirements and Plans to the Team.	Input: Project Plan Plan Details (Detailed Design, Software Construction, Testing and Certification, Training, Conversion, Installation, Implementation) Programmer's Handbook Output: Team Awareness
5.4	Perform Construction of Components Responsible – Developer	Description/Purpose The Detailed Design phase was able to formulate a decomposed technical design of the system. The next step is to transform the design into computer programs. This task produces running and documented programs.
	Subtasks: 1. Identify Next Component for Construction 2. Walk-through Specification with Programmer 3. Translate Component Design to Codes 4. Check Component Source Code for Compliance to Standards 5. Prepare Test Data 6. Test Component 7. Report Status of Component	Input: Function Specification Program Specification Programmer's Handbook Output: Tested Component
5.5	Perform Final System Test Responsible –Developer	Description/Purpose System test is the testing of programs in its environment. This would include the program and all other components that interface with the program. Because programs process input from prior programs and pass their output to subsequent programs, all interfaces are tested. System testing must also include cyclic testing in order to run period-end functions.
	Subtasks: 1. Perform Database Functional Test 2. Perform System Functional Test 3. Conduct Capacity Test 4. Perform Cyclic Test 5. Perform System Performance Test	Input: Test Plan Test Data Output: System Test Results Installation Package
5.6	Complete System Documentation and User Manuals Responsible –Developer	Description/Purpose In the process of program construction and testing, documentation deliverables for the system are also updated.
	Subtasks: 1. Write User Manual using Requirements Definition Manual as basis 2. Compile System Documentation Requirements into one Manual	Input: Documentation in Programs Functional Specifications Database Design Requirements Definition Manual Output: User Manual System Documentation

5.0 Software Construction Activities		
5.7	<p>Complete Certification Test Cases</p> <p>Responsible –Developer or User</p>	<p>Description/Purpose This task is primarily to ascertain that the third party or user certification materials will be in place when needed.</p>
	<p>Subtasks: 1. Complete Certification Test Cases</p>	<p>Input: Test Plan Output: Test Data Certification Test Data</p>
5.8	<p>Develop Training Materials</p> <p>Responsible –Developer</p>	<p>Description/Purpose A training program should be designed for the personnel selected and assigned to the system. Additional presentation materials may be developed to suit the level of all concerned personnel (from managers to involved personnel).</p>
	<p>Subtasks: 1. Identify/Classify Training Requirements 2. Identify Training Media 3. Prepare Course Outline 4. Develop Training Materials</p>	<p>Input: User Manual System Documentation Manual Training Plan Output: Training Materials</p>

Phase 6. Software Certification

The purpose of certification is to assure that the system meets the requirements agreed upon after the Requirements Definition phase. This assurance is achieved by an Alpha Test and a Beta Test.

An Alpha Test checks the system under a typical user environment. This phase may be contracted to a party other than the users. Another approach is to have the test done by the user team and then evaluated by a third party. The need for the Alpha Test depends on the sensitivity of the system’s functions and environment.

A Beta Test is a full product test involving a client’s use of the product. The purpose for a Beta test is to determine client assessment of the product, its performance, installation procedure, documentation, and reliability. This Test is usually done for packages, which are intended for a wide market.

6.0 Software Certification Activities		
6.1	Perform Alpha Certification Testing Responsible – User/Third Party	Description/Purpose Alpha testing provides an internal, project-independent assessment of the correctness, performance and reliability of the system. System validity is determined by verifying requirements, specifications and source program implementation. Program requirements and specifications are reviewed first to establish a base for the testing of the operational code. The degree of testing conducted is dependent upon the thoroughness of the requirements, specifications and program verification procedures. Unnecessary and redundant testing of certified software components may be skipped.
	Subtasks: 1. Finalize Test Plan and Procedures 2. Review and Approve Test Plan 3. Perform Alpha Test	Input: Approved Requirements Definition Manual Certification Plan User Manual System Documentation Manual Certification Test Data Installation Package Output: Approved Alpha Certification Test Plan Alpha Test Results
6.2	Conduct Alpha Test Review and Respond Accordingly Responsible – Developer	Description/Purpose The purpose of this task is to assess the Alpha Test results and respond to the problems found.
	Subtasks: 1. Review Alpha Test Results 2. Identify Alpha Test Action Items 3. Correct/Comply with Action Items 4. Document Response to Action Items	Input: Alpha Test Results Output: Corrected Action Items

6.0 Software Certification Activities		
6.3	<p>Perform Beta Certification Testing</p> <p>Responsible – User</p>	<p>Description/Purpose The purpose of a Beta Test is to determine user reaction to the product.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Finalize Test Plan and Procedures 2. Review and Approve Test Plan 3. Perform Beta Test 	<p>Input: User Manual System Documentation Manual Certification Test Data Installation Package</p> <p>Output: Approved Beta Certification Test Plan Beta Test Results</p>
6.4	<p>Conduct Beta Test Review and Respond Accordingly</p> <p>Responsible – Developer</p>	<p>Description/Purpose The purpose of this task is to assess the Beta Test results and respond to the problems found.</p>
	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Review Beta Test Results 2. Identify Beta Test Action Items 3. Correct/Comply with Action Items 4. Document Response to Action Items 	<p>Input: Beta Test Results</p> <p>Output: Corrected Action Items</p>

Phase 7. Installation

This phase is concerned with placing the certified system into operation. This involves creation of records, training management and operations personnel, installing the system and implementing procedures. Installation end when the customer accepts the system.

As in the other phases, a plan that would direct the remaining activities is established. The plans must have been started after the Detailed Design phase but has to be updated prior to the actual installation.

7.0 Installation Activities			
7.1	<p>Finalize Installation / Conversion / Training Plan</p> <p>Responsible – Developer</p>		
	<p>Description/Purpose The installation/training/conversion team is responsible for identifying potential problem areas, available solutions, responsibilities, available resources and control methods.</p> <p>Potential problem areas can be identified by examining:</p> <ul style="list-style-type: none"> - areas that gave problems during testing - man/machine interfaces - machine interfaces from different providers - new features of the system - schedules 		
	<table border="0" style="width: 100%;"> <tr> <td style="width: 50%; vertical-align: top;"> <p>Subtasks:</p> <ol style="list-style-type: none"> 1. Determine Potential Problem Areas 2. Prepare Plan to Handle Problem Areas 3. Establish Communication Channels 4. Prepare Schedules </td> <td style="width: 50%; vertical-align: top;"> <p>Input: Problem Reports during Testing Original Installation Plan, Training Plan and Conversion Plan</p> <p>Output: Installation Plan Conversion Plan Training Plan</p> </td> </tr> </table>	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Determine Potential Problem Areas 2. Prepare Plan to Handle Problem Areas 3. Establish Communication Channels 4. Prepare Schedules 	<p>Input: Problem Reports during Testing Original Installation Plan, Training Plan and Conversion Plan</p> <p>Output: Installation Plan Conversion Plan Training Plan</p>
<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Determine Potential Problem Areas 2. Prepare Plan to Handle Problem Areas 3. Establish Communication Channels 4. Prepare Schedules 	<p>Input: Problem Reports during Testing Original Installation Plan, Training Plan and Conversion Plan</p> <p>Output: Installation Plan Conversion Plan Training Plan</p>		
7.2	<p>Conduct Management and Operations Training Programs</p> <p>Responsible – Developer</p>		
	<p>Description/Purpose The Management Training Program aims to orient those client managers who will be affected by the new system. They must understand the impact of the system on their areas of responsibility.</p> <p>The Operations Training Program aims to train affected operations personnel on the use of the system.</p>		
	<table border="0" style="width: 100%;"> <tr> <td style="width: 50%; vertical-align: top;"> <p>Subtasks:</p> <ol style="list-style-type: none"> 1. Conduct Management Training <ol style="list-style-type: none"> a. Conversion b. Installation c. Communication Procedure 2. Conduct Operations Training </td> <td style="width: 50%; vertical-align: top;"> <p>Input: Management and Operations Training Plan Training Materials</p> <p>Output: Oriented Management and Operations Personnel</p> </td> </tr> </table>	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Conduct Management Training <ol style="list-style-type: none"> a. Conversion b. Installation c. Communication Procedure 2. Conduct Operations Training 	<p>Input: Management and Operations Training Plan Training Materials</p> <p>Output: Oriented Management and Operations Personnel</p>
<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Conduct Management Training <ol style="list-style-type: none"> a. Conversion b. Installation c. Communication Procedure 2. Conduct Operations Training 	<p>Input: Management and Operations Training Plan Training Materials</p> <p>Output: Oriented Management and Operations Personnel</p>		
7.3	<p>Perform System Conversion</p> <p>Responsible – Developer/User</p>		
	<p>Description/Purpose System conversion involves data, forms, physical facilities, database and other system aspects.</p>		
	<table border="0" style="width: 100%;"> <tr> <td style="width: 50%; vertical-align: top;"> <p>Subtasks:</p> <ol style="list-style-type: none"> 1. Review Items Needed for Conversion 2. Commence Installation and Conversion 3. Convert Physical Facilities 4. Convert Database 5. Convert Forms </td> <td style="width: 50%; vertical-align: top;"> <p>Input: Installation Plan Conversion Plan</p> <p>Output: Final Conversion Report</p> </td> </tr> </table>	<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Review Items Needed for Conversion 2. Commence Installation and Conversion 3. Convert Physical Facilities 4. Convert Database 5. Convert Forms 	<p>Input: Installation Plan Conversion Plan</p> <p>Output: Final Conversion Report</p>
<p>Subtasks:</p> <ol style="list-style-type: none"> 1. Review Items Needed for Conversion 2. Commence Installation and Conversion 3. Convert Physical Facilities 4. Convert Database 5. Convert Forms 	<p>Input: Installation Plan Conversion Plan</p> <p>Output: Final Conversion Report</p>		

7.0	Installation Activities	
7.4	Install System Responsible – Developer	Description/Purpose System installation involves two key factors. One is establishing that the system will fit into the hardware. Another factor is verifying that the system functions according to its specifications.
	Subtasks: 1. Install System.	Input: Installation Package Output: Installed System
7.5	Certify System Responsible – User	Description/Purpose The purpose of this task is to ensure that all elements have been delivered.
	Subtasks: 1. Review System Deliverables 2. Prepare Acceptance Report	Input: Installed System User Manual Documentation Manual Output: User Acceptance Report