GOVERNMENT ICT STANDARDS

Systems and Applications Standard

Second Edition 2019

ICTA.6.002:2019

The ICT Authority is a State Corporation under the State Corporations Act 446
www.icta.go.ke

REVISION OF ICT STANDARDS

In order to keep abreast of progress in industry, ICTA Standards shall be regularly reviewed. Suggestions for improvements to published standards, addressed to the Chief Executive Officer, ICT Authority, are welcome.

ICT AUTHORITY (ICTA)

Head Office: P.O. Box 27150, Nairobi-00100, Tel.: (+254 202) 211 960/61
E-Mail: standards@ict.go.ke, Web:http://standards.icta.go.ke

DOCUMENT CONTROL

Document Name:	Government Systems and Applications Standard
Prepared by:	Government Systems and Applications Technical Committee
Edition:	Second Edition
Approved by:	Board of Directors
Date Approved:	13th January 2020
Effective Date:	1st February 2020
Next Review Date:	After 3 years

Contents

FOREWORD

The ICT Authority has the mandate to set and enforce ICT standards and guidelines across all aspects of information and communication technology including Systems, Infrastructure, Processes, Human Resources and Technology for the public service. The overall purpose of this mandate is to ensure coherent and unified approach to acquisition, deployment, management and operation of ICTs across the public service in order to achieve secure, efficient, flexible, integrated and cost effective deployment and use of ICTs.

To achieve this mandate, the Authority established a standards committee to identify the relevant standard domains and oversee the standards development process. The committee consulted and researched broadly among subject matter experts to ensure conformity to acceptable international and national industry best practices as well as relevance to the Kenyan public service. The committee eventually adopted the Kenya Bureau of Standards (KEBS) format and procedure for standards development. In an engagement founded on a memorandum of understanding KEBS, participated in the development of these Standards and gave invaluable advice and guidance.

For example, the Systems and Applications Standard, which falls under the overall Government Enterprise Architecture (GEA), has therefore been prepared in accordance with KEBS standards development guidelines which are, in turn, based on the international best practices by standards development organizations including ISO.

The Authority's Directorate of Programmes and Standards has the oversight role and responsibility for management, enforcement and review of this standard. The Directorate shall carry out quarterly audits in all the Ministries, Counties, and Agencies (MCA) to determine compliance to this Standard. The Authority shall issue a certificate for compliance to agencies upon inspection and assessment of the level of compliance to the standard. For non-compliant agencies, a report detailing the extent of the deviation and the prevailing circumstances shall be tabled before the Standards Review Board who shall advise and make recommendations to remedy the shortfall.

The ICT Authority management, conscious of the central and core role that standards play in public service integration, fostering shared services and increasing value in ICT investments, shall prioritize the adoption of this standard by all Government agencies. The Authority therefore encourages agencies to adhere to this standard in order to obtain value from their ICT investments.

Dr. Katherine W. Getao, EBS

Chief Executive Officer

ICT Authority

1.0 INTRODUCTION

The complexity of software systems has led to new opportunities but in equal measure increased challenges for organizations that create and utilize systems. These challenges exist throughout the life cycle of a system and at all levels of architectural detail.

This document describes standards for developing (or purchasing and installing) and maintaining/managing computer applications for administrative purposes in Government. The degree to which the responsibility for development, implementation and maintenance of systems is centralized in a single administrative ICT department versus decentralized and handled by the functional offices varies from organization to organization. While these standards will sometimes refer to the "ICT department", these standards apply to any department or any vendor engaged by the MCDAs that undertakes development, installation or maintenance of ICT applications. The determination for when these standards apply depends on the nature of the application, not on who is responsible for the development.

There are two main characteristics of applications which must be considered when determining how these standards apply:

1. The size and complexity of the application; and

2. Some measurement of how essential the system is to the operation of the MCDA or the Government as a whole, and therefore how much risk is associated with whether or not the development efforts are successful.

A system should be considered high risk if a failure of the system to function correctly and on schedule could result in a major failure by the MCDA to perform essential functions, a significant loss of funds to the MCDA, or a significant liability or legal exposure to the MCDA.

For the purposes of determining, a small, less complex system is one that would take less than 1 year of effort (staff time) to develop and implement, or less than Ksh.15,000,000 in cost.

For the purposes of determining the maintenance of a system, a small, less complex maintenance project is one that would take less than 1 month of effort (staff time) to develop and implement, or less than Ksh1,500,000 in cost.

2.0 SCOPE

Overview
This document establishes a common framework for software life cycle processes, with well-defined terminology, that can be referenced by the MCDAs.
The document applies to the acquisition, supply, development, operation, maintenance, and disposal (whether performed internally or externally to the MCDA) of software systems, products and services, and the software portion of any system, Software includes the software portion of firmware.

2.1 Field of Application

This standard applies to:
- Central Government of Kenya
- County Governments
- Constitutional Commissions
- State Corporations

2.2 Targeted Groups

The document defines three target groups for the Government services

a. Government to Citizens (G2C): services which the government offers its citizens directly
b. Government to Business (G2B): services which the government offers to companies
c. Government to Government (G2G): government services for public agencies.

2.3 Domains to be covered

2.3.1 Architectural model for e-government applications

- The enterprise viewpoint
- The information viewpoint
- The computational viewpoint
- The engineering viewpoint
- The technology viewpoint

2.3.2 Application and system software acquisition, maintenance and disposal

- Planning, Analysis and design process
- Application and System software acquisition
- Application and System software development
- Application and System software Maintenance
- Application and System software Retirement
- Application and System software Disposal

2.3.3 Messaging and Collaboration

- E-mail and instant messaging policy
- Video and audio conferencing policy
- Social media use policy

2.3.4 Websites

- Web Governance
- Domain management
- Web design
- Web Branding
- Web Hosting
- Web Content

2.3.5 Interoperability

2.3.6 Integration

2.3.7 Licensing

2.3.8 Governing of systems

2.4 Limitations

This document does not prescribe a specific software life cycle model, development methodology, method, modeling approach, or technique. The users of this document are responsible for selecting a life cycle model for the project and mapping the processes, activities, and tasks in this document into that model. The parties are also responsible for selecting and applying appropriate methodologies, methods, models and techniques suitable for the project.

3.0 NORMATIVE REFERENCES

The following standards contain provisions which, through reference in this text, constitute provisions of this standard. All standards are subject to revision and, since any reference to a standard is deemed to be a reference to the latest edition of that standard, parties to agreements based on this standard are encouraged to take steps to ensure the use of the most recent editions of the standards indicated below. Information on currently valid national and international standards can be obtained from Kenya Bureau of Standards

- ISO/IEC 12207:2017 Systems and Software Engineering - Software lifecycle processes

- ISO 90003:2018 Software engineering Guidelines

4.0 ABBREVIATIONS

SDLC	Software Development Lifecycle
SCMC	Software Change Management Committee
MCDA	Ministries, Counties, Department and Agencies
CMM	Capability Maturity Model
CD-ROM	Compact Disc Drive
IM	Instant Messaging
NIST	National Institute of Standards and Technology
SDLC	Software Development Lifecycle
COBIT	Control Objectives for Information Technology
HTML	Hypertext Markup Language
XHTML	Extensible Hypertext Markup Language
XML	Extensible markup language
CSS	Cascading style sheets
SEO	Search Engine Optimization
MCDA	Ministries Counties and Agencies
G2C	Government to Citizen
G2B	Government to Business
G2G	Government to Government
CMS	Content Management System
ICT	Information Communication Technologies
COTS	Commercial off the shelf software

5.0 TERMS AND DEFINITIONS

For the purposes of this Kenya Standard, the following definitions apply:

5.1 Digital Asset

A digital asset is any form of content and/or media that have been formatted into a binary source which include the right to use it. A digital file without the right to use it is not an asset.

5.2 Systems software

System software (systems software) is computer software designed to operate and control the computer hardware and to provide a platform for running application software. System software can be separated into two different categories, operating systems and utility software.

5.3 Application software

Applications software (also called end-user programs) include such things as database programs, word processors, Web browsers and spreadsheets

5.4 Application development software

Is the software used for computer programming, documenting, testing, and bug fixing involved in creating and maintaining applications and frameworks involved in a software release life cycle and resulting in a software product.

5.5 Open source software

Open source software is computer software with its source code made available with a license in which the copyright holder provides the rights to study, change, and distribute the software to anyone and for any purpose
.

5.6 Application

Computer programs, procedures, rules, and associated documentation and data pertaining to the operation of a computer system.

5.7 Computer

A programmable machine that receives input, stores and manipulates data/information, and provides output in a useful format.

5.8 Confidentiality

The security goal that generates the requirement for protection from intentional or accidental attempts to perform unauthorized data reads. Confidentiality covers data in storage, during processing, and while in transit.

5.9 Data

Groups of information that represents the qualitative or quantitative attributes of a variable or set of variables. Data are often viewed as the lowest level of abstraction from which information and knowledge are derived.

5.10 Data Integrity

The property that data has not been altered in an unauthorized manner. Data integrity covers data in storage, during processing, and while in transit.

5.11 Email

Messages, usually text, sent from one person to another via electronic medium. Email may also be sent automatically to a large number of addresses (mailing list).
and which sets out their approach to managing information security objectives.

5.12 Process

A structured set of activities designed to accomplish a specific objective

5.13 Resource

A Governments physical or virtual entities (human or otherwise) that are of limited availability and can be used to undertake operations or business change.

5.14 SDLC

Software Development Lifecycle - A structure imposed on the development of a software product. The SDLC is a systematic approach to the creation of software or application. This cycle typically includes a requirement analysis, design, coding, test, implementation and post-implementation phases.

5.15 Software

The collection of computer programs and related data that provide the instructions telling a computer what to do.

5.16 Web Systems Administrator

Maintains the systems environment of the website by identifying system requirements; selects, installs and configures server hardware, software and operating systems, installs upgrades, defines system and operational policies and procedures, assesses access information and security requirements and monitors system performance.

5.17 Web Designer / Developer

Responsible for the design, layout and coding of a website. Involved with the technical and graphical design aspects of a website – how the site works and how it looks. They may also be involved with the maintenance and update of an existing site. A person who deals only with the graphical and appearance elements would be a web designer, while the one who focuses on coding is a web developer. These roles are often combined.

5.18 Web Content Manager

A web content manager updates websites, blogs and other sites that require regular update. The person is responsible for editing, posting and removing content from the site. The person may or may not be responsible for producing the actual content.

5.19 Web Master

Small companies sometimes employ a webmaster who is responsible for all the job roles described above. A webmaster is also sometimes the role given to a senior person to establish the overall corporate Web design and policies, arrange all the necessary technical resources, and supervise the design of the corporate website.

5.20 Extensible Hypertext Markup Language (XHTML)

XHTML is part of the family of XML markup languages. It mirrors or extends versions of the widely used Hypertext Markup Language (HTML), the language in which Web pages are formulated.

5.21 Privacy

Privacy, in this context, is defined as the right of individuals to know the personal information that is maintained about them and to control the ways in which that information is collected, disseminated, maintained and used.

6.0 DETAILED STANDARDS FOR THE DOMAINS

6.1 Architectural model for e-government applications

- The enterprise viewpoint (Annex 1)
- The information viewpoint (Annex 2)
- The computational viewpoint (Annex 3)
- The engineering viewpoint (Annex 4)
- The technology viewpoint(Annex 5)

6.2 SOFTWARE ACQUISITION, MAINTENANCE AND DISPOSAL

This sections covers standards for developing (or purchasing and installing) and maintaining computer applications in the whole of Government. The degree to which the responsibility for development, implementation and maintenance of systems is centralized in a single administrative ICT department versus decentralized and handled by the functional offices varies from organization to organization. While these standards will sometimes refer to the "ICT department", these standards apply to any department or any vendor engaged by the MCDAs that undertakes development, installation or maintenance of ICT applications. The determination for when these standards apply depends on the nature of the application, not on who is responsible for the development.

6.2.1 Planning, Analysis and design process

6.2.1.1 Project Planning and Management

Project management is a task which spans across all phases of systems development. The Project Plan, a key tool for effective project management. A Project Plan, used as part of the management of a project, can reduce the likelihood of major, unexpected schedule or cost overruns.

6.2.1.1.1 Project plan

a) Each project must have a plan to guide its implementation unless an exception is granted by the administrative head in charge of computing department or equivalent.

b) The plan must be updated regularly throughout the development phases to reflect any changes or refinements encountered.

6.2.1.1.2 Staff Time Estimates

a) Estimates for staff time should be developed by the completion of the System Definition phase (Prototyping Track) or Requirements Definition phase (Traditional or Purchase Track), although these estimates may be adjusted as needed during the Design, Prototyping, or Installation phases.

b) If on-going support for the system after completion of the project is expected to be significant, then estimates of the required staff time for on-going support should be made.

c) Estimates for functional office staff time may also be included in the Project Plan. It is important that adequate resources in terms of functional office staff time are committed to the project.

d) Time required for functional office staff training should also be considered

.

6.2.1.1.3 Project scope

a) The MCDA shall establish the overall scope of the project early in the development cycle.
b) The project leader shall be the custodian of the approved project scope
c) If major issues arise regarding the scope of the project, then senior level management may need to become involved to determine whether the scope should be expanded significantly (with a corresponding expansion in schedule and resources), or remain at the original level.

6.2.1.1.4 Project status reporting

a) As part of the management of projects, regular status reports must be provided to the Accounting Officer by the Project Leader for each significant development project.

The following should be addressed in the report;
i. What has been completed since the last report;
ii. Progress on meeting major milestones, plus any changes in estimated completion dates for future milestones;
iii. Any issues that have arisen, such as difficulties in reaching agreement among the users of the system on a design issue, or a recently discovered interface file problem; and
iv. Review of the remaining tasks and the next milestone, together with the estimates for staff time required.

b) Project status reports shall be retained by the accounting officer.

6.2.1.1.5 Legal framework

The MCDA shall define and state the legal framework where the system that is being developed is anchored.

6.2.1.1.6 Roles and responsibilities

The MCDA shall identify the participating departments, entities and clarify responsibilities.

Assignment of responsibility for systems development must be accompanied by a corresponding commitment of staff time from each participating office.
While defining responsibilities, MCDA shall identify a single entity and assign clear ownership or sponsorship of the project. This person or group will be responsible for making key decisions, such as determining whether to increase the scope or budget of the project during development if changes arise, or to stay within the originally-allocated budget and schedule and forego the proposed changes.

MCDA shall agree on the procedure of handling changes that may arise during the systems development process as part of the determination of responsibilities.
NOTE: The possible split of responsibilities between the functional office (or offices) and the ICT department varies along a continuum from the ICT department taking primary responsibility throughout the project development effort, to the functional office taking primary responsibility with the ICT department serving in the role of a consultant.

6.2.1.1.7 Steering Committee and Project team

For larger projects, MCDA shall form a high-level steering committee composed of senior-level management from functional offices and the ICT department. The steering committee will be responsible for;

a) providing overall strategic direction and decisions for the project
b) establishing the business case for system development and the planning principles
c) defining the overall scope of the project
d) communicating with all stakeholders in the system
e) developing a decision framework
f) and providing management oversight throughout development and implementation.

A Project Leader shall be appointed who will be responsible for the day-to-day management and oversight of the project
.

6.2.1.1.8 Internal Audit

Internal Audit's involvement during high risk systems development should be conducted.
The level of involvement for non-high risk applications will be determined by Internal Audit.

6.2.1.2 Analysis and Design

The development of new systems or major enhancements to existing systems is often the result of significant changes made to the business processes supported by the systems.
Usually the effort to simplify the business processes themselves precedes any major systems development effort. It is appropriate that the business processes be reviewed before systems work begins, to avoid the unfortunate mistake of simply automating existing cumbersome processes.
Ideally the efforts to simplify business processes will be done by the functional office in conjunction with technical personnel, so that current technology can be considered as the business processes are reviewed. In some cases, particularly when a vendor package is selected for implementation, the simplification of business processes may occur during the systems development or installation process.

The project team must undertake to:

1. Analyse business and system requirements

2. The specific intended use of the system to be developed / acquired must be analysed to specify system requirements. The system requirements specification should describe: functions and capabilities of the system; business, organizational and user requirements; safety, security, information, privacy, interface, operations, and maintenance requirements; design constraints and qualification requirements.

3. The system requirements specification must be documented.

4. A top-level architecture of the system must be established. The architecture should identify items of hardware, application/software, and manual-operations. It should be ensured that all the system requirements are allocated among the items.

5. The system architecture and the system requirements allocated to the items must be documented.

6.2.2 APPLICATION AND SYSTEM SOFTWARE ACQUISITION

Policies regarding acquisition of hardware and software must be followed and appropriate approvals obtained.
Application software, systems software and application development software shall be acquired in consideration of information sharing, user satisfaction, compatibility, unified support, Interoperability, scalability, quality and improved staff productivity. The MCDA will be required to determine whether to develop internally or externally or vendor packages.
In acquiring information systems, MCDAs shall to the greatest possible extent develop, create and procure software based on the use of open standards.

6.2.2.1 Procurement

When procuring software, the MCDA shall;

a) Ensure that there are no already existing software applications within Government that provides equivalent functions and that can be replicated in the organization before procuring any software to avoid duplication.

b) Ensure that the architectural model for e-government applications is conformed with. (See Annex 1-5).

c) Ensure that acquisition of the software is done with consultation and coordination of the Head of ICT Unit who shall be responsible in the preparation and issuance of all technical specifications for the software, as well as ensuring that the guidelines stipulated herein are adhered to.

d) Use requisition and acceptance forms to ensure that requests for procurement of software are validated by the respective Heads of Department.

e) Ensure that requirements are clearly defined and documented when procuring enterprise software.

f) Where possible, MCDAs are required to use enterprise version of software.

g) Software will only be installed by ICT officers in MCDAs. Users shall not be authorized to install any software on computers.

h) Procure and use the latest version of software. Where a previous version of software is to be used, MCDAs shall be required to give justification to the ICT Steering Committee as per the IT governance standard.

6.2.2.2 In house Development

When developing software in house, MCDA shall;

a) Adopt a project management approach.
b) Ensure that an optimal system development methodology such as software development lifecycle is adopted in order to obtain a useful system.
c) Constitute a development team consisting of various specializations as may be required in specific software development task. These shall include software developers with expertise in target development platform, business/systems analysts, business/systems designers, database experts, network and communication, security specialist among other skills that may be required in different project.
d) Establish and maintain an intellectual property agreement on software developed in-house.
6.2.2.3 Outsourced Development.

For sophisticated systems development initiatives, the MCDA may contract an external developer to deliver the business application.

MCDA shall consider the following tasks;

a) Define a strategy on how acquisition will be conducted.
b) Prepare a request for the supply of a product or service that includes the requirements.
c) Communicate the request for the supply of a product or service to potential suppliers
d) Select one or more suppliers.
e) Develop an agreement with the supplier that includes acceptance criteria.
f) Identify necessary changes to the agreement.
g) Evaluate impact of changes on the agreement.
h) Negotiate the agreement with the supplier.
i) Update the agreement with the supplier, as necessary.
j) Assess the execution of the agreement.
k) Provide data needed by the supplier and resolve issues in a timely manner.
l) Confirm that the delivered product or service complies with the agreement.
m) Provide payment or other agreed consideration.
n) Accept the product or service from the supplier, or other party, as directed by the agreement.
o) It should be ensured that source code ownership is defined before engagement with the developer.
p) Close the agreement.

6.2.2.4 Commercial Off the shelf

If the MCDA decides to acquire a specific commercial off the shelf software or open source software, acquisition can be limited to identifying the supplier, accepting or negotiating the conditions in a pre-defined license or lease or maintenance agreement, determining rights to intellectual property and data rights in the software system, and agreeing on the price.

A factor to be considered by MCDAs in agreements between suppliers is data rights and lateral access to constituent data and intellectual property. As an example, suppliers for one system component may need to collaborate with suppliers for another component and share source code. Agreements can enable this collaboration.

6.2.2.5 Software Leasing

When the option to purchase and fully own a software product that meets an MCDAs'business requirements is not feasible, the agency can lease a market product that is deemed to fully satisfy the business requirements of the organization (MCDA). if this happens to be so, It should be ensured that;

1. Before the decision to lease the software product is reached, The MCDA must have documented justification (signed by the accounting officer) on why leasing is the only option.

2. The MCDA may seek oversight from ICTA during the acquisition and implementation of the leased Application.

3. The MCDA shall be the custodian of both raw and processed data; The Vendor has no right over the data captured by the leased software.

4. Both the raw and processed data must be accessible, managed and retained by the MCDA

5. In the lease agreement, data access procedures should be defined; the agreement must detail how the MCDA Data shall be accesed on lease expiry.

6. Appointed MCDA ICT staff maintaing the system must be trained on methods of data extraction from the lease product after expiry of the lease period.

For this standard, the undertanding is that a leased software does not belong to the MCDA and the MCDA can not claim ownership of the software product except the data it stores.

6.2.2.6 System Software licenses

Different systems are implemented with varying licensing models; MCDAs shall ensure that;

a) Licenses for commercial operating system are provided upon acquisition, duly registered and subsequently renewed as per the requirements of the copyrights

b) Service level agreements are signed with the vendors.

c) The latest stable version is purchased in each case.

d) Users are trained on any new client operating system software.

6.2.3 Application and System software development

This document provides standards and guidelines for the systems development process. This document does not provide standards for reviewing the functional offices' business processes, but it is recognized that this review will frequently precede or may coincide with the systems development process.

6.2.3.1 General Consideration

MCDAs shall take into consideration the following when acquiring application development software:

i. Type of application to be developed; Desktop application, Web-based application or server application.
ii. Operating System platform the software to be developed is to run on.
iii. Integration with the existing systems.
iv. Database to be used by the application.
v. Compatibility with existing and future hardware and software platforms.
vi. The Speed of development.
vii. Performance of compiled code.
viii. Assistance in enforcement of code
ix. Portability; can the application developed be used in an operating systems other than the one in which it was created without requiring major rework.
x. Fitness of the software for the application being developed.

6.2.3.2 Skills

MCDAs shall ensure that ICT officers responsible for development of software are adequately trained on all application software acquired.

6.2.3.3 Documentation of systems

MCDAs shall ensure that all systems have the following documentation;

a) Project initiation documentation detailing the business case
b) Feasibility study detailing the proposed solution
c) Detailed user and technical requirements
d) High level and detailed system design documents
e) System testing and commissioning documentation
f) Evidence of user and technical training
g) User and technical manuals
h) Certificate of completion

6.2.3.4 Systems Development Tracks

The exact methods employed for systems development will vary depending on the specific project. Although every systems project is unique, there are three key characteristics which will influence the overall approach chosen for systems development:

1. The overall size and complexity of the application;
2. The technology to be used for developing the application; and
3. Whether the system will be a purchased package, or custom developed.

Clearly, a smaller, less complex application will most likely not require as many separate phases as a larger, more complex application. In addition, some of the newer development technologies such as GUI-based development tools allow a different approach than a more traditional technology such as 3GL languages. Finally, if the system will be a purchased vendor package, then the phases of the development process will differ from those for a system being developed from scratch.

Three separate development approaches or "tracks" have been identified:

a) The first, involving prototyping, is appropriate for custom development of smaller systems or systems that use newer technology such as GUI-based development tools;
b) The second, involving a more traditional life cycle approach, is more suited to custom development of larger systems using 3GL/4GL languages; and
c) The third is tailored for the purchase and implementation of vendor packages.

Some phases of project development apply to two or all three approaches, whereas others are unique to one approach or another. The phases in these three approaches, or tracks, are listed below:

Track 1: Prototyping	Track 2: Traditional Life Cycle	Track 3: Vendor Package Purchase
Project Proposal	Project Proposal	Project Proposal
		Request for Information
System Definition	Requirements Definition	Requirements Definition
		Request for Proposal
Feasibility Study	Feasibility Study	Feasibility Study
		Vendor Contract and Installation Plan
Prototyping	General Design	
	Detail Design	
	Programming/development and Unit Testing	
System Testing	System Testing	System Testing
Implementation	Implementation	Implementation
Final Documentation	Final Documentation	Final Documentation

Post-Implementation Review	Post-Implementation Review	Post-Implementation Review

There are several important points to be noted regarding the phases listed above for each of the three approaches to system development:

1. The development of any one system may not fall neatly into one of the three categories listed above.

2. Depending on the individual project, not every phase may be needed. For example, when developing a custom-built application where there is general agreement as to the overall approach, there most likely will be no need for a Feasibility Study.

3. Although the phases listed above are listed in the general sequence in which they occur, it is important to note that phases will overlap. For example, Final Documentation must be completed at about the same time as implementation of the system. To complete the documentation by the time the system is implemented (or, shortly thereafter), the documentation must be started much earlier in the development process.
The above phases for each of the three tracks are described in Annex 6, Phases of Systems Development.

6.2.3.4.1 Prototyping Track.

As noted above, the prototyping methodology is best suited to developing systems (or portions of systems) where:

a) The system is small to medium in size;

b) The technology to be used for development (e.g., GUI-based development tools) is not amenable to the traditional approach to systems development normally used with 3/4GL systems development; and/or the system involves heavy interaction with the functional office users (e.g., online processing).

Since prototyping is based upon a high degree of involvement by the functional office in the specification of system functions, it is most useful for developing system functions with which end users will interact directly, such as online systems or online portions of systems. Also, this method works best when a prototype may be developed in a relatively short period, and the time between iterations of the working model is not overly long. For this reason, the method is best used for small to medium scale systems, or small to medium scale portions of larger-scale systems. This track is also appropriate for systems or portions of systems where a Joint Application Development (JAD) or Rapid Application Development (RAD) methodology will be employed.

6.2.3.4.2 Traditional Life Cycle Track.

The Traditional Track relies more heavily than the Prototyping Track on formal written specifications, particularly for the design phases specifying the programming to be performed. This Track is appropriate for systems where:

a) The system is large, and multiple application developers will be working simultaneously on the project, particularly during the programming and unit testing phase;

b) The system will use 3GL technology, which requires application developers to explicitly code more than may be necessary with 4GL or GUI-based development tools;

c) The system has little direct interaction with the users (e.g., much of the processing is batch); and/or

d) It is difficult for the functional office(s) to confirm the correctness of the programming directly due to the complexity of the logic and/or the nature of the processing, making written specifications vital for functional office review (e.g., in the case of a complex calculation).

It is particularly important with a large system development project involving many people in design, programming and testing efforts to have written design documents, to help prevent misunderstandings among the application developers.

6.2.3.4.3 Vendor Package Purchase and Installation Track

Purchasing and installing a vendor package rather than developing a system from scratch can save significant time, and often significant resources, in terms of cost. There are many issues to consider when deciding whether to purchase a product or develop a custom system, but the following guidelines help determine if purchase of a vendor package should be considered:

a) Whether packages are available on the market that can satisfy at least 80% of the functional requirements of the system.

b) While it may not be possible to find a package that exactly matches the specific functional requirements, if less than 80% of the requirements are satisfied then purchase of a package is likely to be more costly than custom development.

c) If packages available on the market that meet the functional needs adequately are in production, not planned for future release.

d) If packages are available which are compatible with the technical environment currently available at the MCDA.

e) Whether the overall cost of purchasing and installing (and maintaining) a package will be at least no more than (and preferably less than) the cost of custom development and maintenance.

f) Whether or not the vendors for the packages available have a proven track record of installation and support and can show evidence of financial stability (this is particularly critical for large systems).

6.2.3.4.4 Selection of One or More Tracks.

Most systems will use one track as the primary track, but some systems may use a combination of two, or possibly even all three, tracks.

One common instance where more than one track might be used involves the installation of a large vendor package. Most likely, interface processes will need to be heavily revised or rewritten when a vendor package is installed.

In addition, some customization of the product may be planned before installation. The development of heavily modified or rewritten interface processes plus any customized modifications may be treated as separate, custom-developed projects, which would follow the phases listed under the Prototyping Track or the Traditional Life Cycle Track, although the larger, overall project would follow the Vendor Package Purchase and Installation Track.

The Project Leader is responsible for making a recommendation to his/her manager regarding which of the development tracks or combination of tracks is the best approach for a project. The manager is responsible for final determination of which track or tracks will be used for a project.

6.2.3.5 Security

The purpose of the Software security is to provide objective evidence that the software achieves satisfactory levels of certainty that sufficient protection is achieved against intentional subversion or forced failure due to the software architecture, design, or construction of the code.

a) The MCDA shall ensure that all critical applications (See appendix 1), are manned by qualified Government ICT staff who have been accredited by ICT Authority.

b) MCDAs shall define requirements for achieving software assurance characteristics.

c) MCDAs shall define and implement approach for achieving the desired software assurance.

d) Continuously monitor the extent of achievement of the security requirements.

e) Specify and achieve a satisfactory level of the critical software assurance characteristics.

f) MCDAs shall ensure that software and applications acquisition process conforms to the information security standard.

6.2.3.6 Application Testing

a) The project team must conduct testing in accordance with the requirements for the application. It must be ensured that the implementation of each application requirement is tested for compliance.

b) For each requirement of the system, a set of tests, test cases (inputs, outputs, test criteria), and test procedures for conducting System Testing must be developed and documented. The project team must ensure that the integrated system is ready for System Testing.

c) The project team MUST produce the following minimum set of documentation as part of this phase:

- Test plan
- Test Criteria
- Test Reports
- Defect/Error report
- Change Requests

d) Upon successful completion of the review(s), a baseline for the testing of the application must be established and formal sign-off for this phase must be obtained.

6.2.4 Application and System Software Maintenance

Application software and systems software shall be maintained to ensure availability of service and business continuity.

6.2.4.1 Systems and Applications Register

6.2.4.1.1 Inventory

The MCDAs shall keep an inventory of all software and give annual reports on status of utilization, support and adaptability with the following attributes;

a) Systems name
b) Systems purpose
c) Supporting technologies
d) Number of users per system
e) Number of ICT support staff
f) Organizational coverage
g) Scope of use
h) Anticipated end-of-life
i) Commercial name

6.2.4.1.2 Documentation

The MCDA shall;

a) Securely store all software media and administration documentation
b) Create and store copies in line with the disaster recovery plan.

6.2.4.1.3 Costs

The MCDA shall manage all software related costs. This includes;

a) The original capital value/ estimated replacement cost of the application.
b) Operational costs
c) Depreciation costs
d) Licensing costs
e) Maintenance costs
f) Development and enhancement costs

g) Annual estimated cost of operation

6.2.4.1.4 Condition of Applications

The MCDA shall prepare a report on the condition of the applications in terms of:

a) Organizational architecture alignment
b) Government Enterprise Architecture (GEA) alignment
c) Integration
d) Authentication
e) Maintainability
f) Portability
g) Scalability
h) Availability
i) Performance
j) Usability

6.2.4.1.5 Future Business Value

The MCDAs shall continually assess the performance of the applications in line with their future business objectives.

6.2.4.2 Change Management

- MCDAs shall establish standards and procedures for changes to application software in conformance with these standards.

- The MCDA shall put in place a Software Change Management Committee (SCMC) to approve all system changes other than vendor supplied patches.
- For changes to Small/Simple, Low Risk systems and application software changes that do not exceed 1 month of effort (IT staff time) to complete or more than $15,000 in total cost, only the following apply:

a) The request for change must be logged and retained. E mail can serve these purposes;
b) Program changes must be documented and dated; and
c) Additional local MCDA maintenance standards for small projects, if any, must be applied.

- For changes to Large/Complex, High Risk systems and for application software changes that exceed 12 months of effort (staff time) to complete or more than KHS 15,000,000 in cost, Phases of Systems Development apply.

- If changes to software are required, the SCMC shall determine;

a) The effect the change will have on the security controls in the software;
b) If consent of the vendor is required;
c) If the required functionality is included in a new version of the software; and
d) If MCDAs will become responsible for maintenance of the software as a result of the change.

- MCDA should provide for the participation of users at appropriate stages of the change process.
- The MCDA shall ensure that all changes are documented and retained.

6.2.4.3 Licenses

The MCDAs shall ensure that the software licenses are maintained to ensure compliance. In the unlikely event of expiry of a software license, the MCDA should remedy the situation and use appropriate tools to the extract the data.
MCDAs shall ensure that there is Service level agreements signed with the vendors.

6.2.4.4 Upgrade

The MCDA shall conduct software updates to ensure that;

a) The most up-to-date approved patches have been applied; and
b) The version of software is vendor supported.

6.2.4.5 Support

a) Software maintenance shall be done in-house by ICT Units who shall develop a maintenance schedule on upgrading and debugging.
b) Sub-contracting for software maintenance shall be through appropriate justification and approval by the Accounting officer. Due diligence shall be undertaken in retaining such contractors.
c) The Head of ICT Unit shall prepare an annual maintenance report and forward it to the Accounting Officer.
d) Software media shall be tagged with the standard government labeling conventions and appropriately physically secured.

6.2.4.6 Security Audit

MCDAs shall ensure that all software and applications are audited annually to ensure they conform with information security standard.

6.2.4.7 Training and Knowledge Transfer

MCDAs shall ensure that ICT officers mandated to maintain or support software acquired are adequately trained.

Where a maintenance contract is in place, MCDAs shall ensure that measures are put in place to enforce knowledge transfer to ICT officers by contractors and vendors for continuous support and maintenance of the system once the contract expires.

6.2.5 Application and System software Transition and Retirement

Purpose

The Transition process is often used for recurring deployments of software to different environments, e.g., from a development environment to a test or maintenance environment, or between various test environments, or from one operational environment to another (e.g, rehosting or use of cloud services). Transitions to backup or contingent sites are typically planned and rehearsed for business continuity and disaster recovery.

a) MCDA shall prepare for and perform system transition to ensure business continuity.
b) MCDA shall define a strategy for managing software releases and other software system transitions.
c) MCDA shall identify and define facility, site, communications network, or target environment changes needed for software system installation or transition.
d) MCDA shall identify information needs and arrange for user documentation and training of operators, users, and other stakeholders necessary for system utilization and support.
e) MCDA shall prepare detailed transition information, such as plans, schedules, and procedures.
f) Identify system constraints from transition to be incorporated in the software system requirements, architecture or design.
g) Identify and plan for the necessary enabling systems or services needed to support transition.
h) MCDA shall manage results of transition.

NOTE: Transition can involve knowledge transfer using the Knowledge Management process.

6.2.6 Application and System software Disposal

The purpose of the Disposal process is to end the existence of a system element or system for a specified intended use, appropriately handle replaced or retired elements, and to properly attend to identified critical disposal needs (e.g., per an agreement, per organizational policy, or for environmental, legal, safety, security aspects).
Application software, systems software and application development software shall be disposed at the end of life in consideration of information security.

The MCDA shall develop and maintain a policy to guide disposal of software. The MCDA shall implement the following activities and tasks in accordance with applicable organization policies and procedures with respect to the Disposal policy process.

6.2.6.1 Preparation for disposal.

The MCDA shall take into account the following:

1) Define a disposal strategy for the software system, to include each system element and to identify and address critical disposal needs, including the following considerations:

a) Permanent termination of the system's functions and delivery of services, e.g., physical destruction of data storage devices, or transition of the software system elements for future reuse in modified or adapted form;

b) Identification of ownership and responsibility for retention or destruction of data and intellectual property in the software system;
c) Transformation of the product into, or retention in a socially and physically acceptable state, thereby avoiding subsequent adverse effects on stakeholders, society and the environment;
d) The health, safety, security and privacy concerns applicable to disposal actions and to the long term condition of resulting physical material and information;
e) Notification to relevant stakeholders of significant disposal activities, e.g., retirement or replacement of a system, software products or services, retirement schedule, or replacement options; and
f) Identification of schedules, actions, responsibilities, and resources for disposal activities.

6.2.6.2 Perform disposal.

In performing the disposal the MCDA shall take into account the following:

a) Deactivate the software system or element to prepare it for removal.
b) Remove the software system, its elements, its data, and non reusable material from use or production for appropriate disposition and action.
c) Withdraw impacted operating staff from the software system or system element and record relevant operating knowledge.
d) Reuse, recycle, recondition, overhaul, archive, or destroy designated software system elements.
e) Conduct destruction of the system elements, as necessary, to reduce the amount of waste treatment or to make the waste easier to handle.

6.2.6.3 Finalize the disposal.

The MCDA shall take into account the following:

a) Confirm that detrimental health, safety, security, and environmental conditions following disposal have been identified and treated.
b) Return the environment to its original state or to a state that is specified by agreement.
c) Archive information gathered through the lifetime of the product to permit audits and reviews in the event of longterm hazards to health, safety, security and the environment, and to permit future software system creators and users to build a knowledge base from experience.

6.3 Messaging and Collaboration

6.3.1 General Considerations

Collaboration systems acquired by an MCDA shall: -

a) Support Features such as email messaging, IP telephony, instant messaging, personal voice service, conference call services, data conference services, document and file sharing, collaborative document and file sharing, forums, data conferencing (sharing of a white board), short message service, chat, internal bulletin, address book, video and single sign-on.
b) Integrate with existing directory systems for access to contact information.
c) Enable grouping of users.
d) Enable a single sign on to all the services.
e) Provide electronic group calendaring and scheduling.

f) Project management systems to schedule, track and charts step in a project as it is being completed.
g) Workflow systems to manage the collaborative flow of documents and tasks.
h) Support Intranet portal integration.
i) Support different client operating platforms.
j) Support common standards for interoperability with collaboration systems in other MCDAs.
k) Support email push to mobile devices.

The collaboration tools shall conform to the following as per the standards:
a) Ease of use
b) Agility
c) Scalability
d) Adaptable contexts
e) Support

6.3.2 Email and Instant Messaging

MCDAs shall acquire and ensure appropriate use and management of E-mail and Instant messaging applications.
The MCDA shall establish and implement a corporate/business email system for official communications.
The MCDA shall acquire e-mail software in accordance to application software acquisition standard.
MCDAs shall ensure that the corporate email software solutions acquired provide for:-

a) Sending of group emails
b) Creation of mailing lists from the server.
c) Email search and retrieve.
d) Creation of email folders.
e) Email archiving.
f) Global address book for all registered users.
g) Sending email attachments.
h) Appending of a Digital Signature.
i) Formatting of e-mail messages (Text formatting, appending of graphics).
j) Email Account management.
k) Security; Real-time spam and Junk mail filtering, password management and client/server system patching
l) Adequate disk quota for all email users.
m) Back up of user mailboxes.
n) Push to email support for mobile devices.
o) The protocols that shall be supported by email solutions acquired by MDA's shall include but not limited to SMTP, MIME, POP3, IMAP4, LDAP version 3, SSL, TLS and Secure MIME.
p) Scalability
q) Compatibility
r) Ensure that the server is protected with a firewall and Antivirus software is installed and regularly updated.
s) Email transmission is secured through the use of encryption technology.

The MCDA shall put in place an email policy to cover the following;

a) Business use of email
b) Personal use of email
c) Privileges and responsibilities
d) Email acquisition, hosting and setup

6.3.3 Video and Audio Conferencing

MCDAs shall ensure appropriate acquisition, management and use of video and audio conferencing applications

6.3.4 Video and Audio Conferencing software

When a video and audio conferencing software is required, MCDA shall ensure the VoIP software acquired provides for;

a) Traditional calling features including call by name, caller ID, last number redial, hold, call waiting, call forwarding, transfer, divert, park, retrieve, voice mail, return call and call conferencing
b) Call Coverage Make it easy to ensure that important calls are answered by administrative assistants or team members, via user-controlled Delegation and Team Calling respectively.
c) Telephone Directory.
d) Maintain Call history.
e) Local Number portability, that is, ability to maintain phone numbers when one changes service providers.

6.3.5 Security

The MCDA shall;

a) Ensure that all VoIP communications and systems are secured.
b) create awareness to users on how to securely use VOIP systems
c) Ensure where possible that end-to-end encryption of the VoIP conversations is employed.
d) Where possible, MCDAs shall Endeavour to separate voice and data traffic logically on the network due to bandwidth, security and Quality of service requirement of VOIP.

6.3.6 Protocols

The MCDA shall ensure that protocols used for video and audio conferencing are in line with provision in the Network standard on VOIP.

6.3.7 Inventory

MCDAs shall document and maintain an inventory of authorized VoIP instruments and shall ensure that the VoIP systems only register and use authorized terminals.

6.3.8 Social Media

a) MCDAs shall ensure appropriate use and management of social media applications
b) MCDA shall establish and maintain a social media policy and create awareness of the policy to its employees.

6.3.8.1 Risk Assessment

a) Prior to authorizing and enabling Internet access to Social Media, MCDA management shall conduct a formal risk assessment of the proposed connections utilizing Risk Management processes.
b) MCDA shall document this risk analysis and retain the documented information.

6.3.8.2 Social Media Management

a) The MCDA shall put in place function to manage MCDA presence on social media networks/ platform.
b) MCDAs shall disseminate rules of engagement on social media.

6.4 Websites Development and Management

6.4.1 Web Governance

MCDAs shall establish a governance structure to manage websites.

6.4.2 Digital Asset Management Committee

A digital asset management committee shall be constituted to:
i. Align the digital asset with public vision, mission and objectives;
ii. Ensure that the digital asset is of high quality and meets its performance targets;
iii. Take responsibility for decisions made about the website.

The committee shall report to the head of the MCDA and Membership of the committee shall include; -

i. Head of Communication (Chair)
ii. Head of ICT (Secretary)
iii. Head of each line function in the MCDA

The committee shall be responsible for; -

i. Drafting of vision, mission and objectives of the organizational digital asset for approval by the Chief Steward.
ii. Recommending the approval of the branding strategy, general template and editorial policy of the digital asset
iii. Setting the performance targets of the digital asset and reviewing performance reports.
iv. Reviewing the budget for the digital asset and recommending its approval by the Chief Steward.
v. Monitoring and evaluating the digital asset, the user feedback that it generates and making recommendations for continuous improvement to the Chief Steward.

vi. Making recommendations to the Chief Steward to develop a new digital asset and / or to retire an obsolete digital asset.

The committee shall meet whenever a major decision about the digital asset is required, but not less than once per quarter (for performance review.)

6.4.3 Digital Asset Technical Committee

A Digital Asset Technical Committee shall be constituted and shall comprise of the following;

i. Head of ICT (Chair)
ii. Public Relation and Communication representative (Secretary)
iii. ICT Security Representative
iv. Web Master
v. Web Administrator
vi. ICT Networking Representative
vii. ICT Database Representative
viii. Information Representative

The responsibilities of DATC shall be;
i. Ensuring that the digital asset complies with technical standards and requirements;
ii. Ensuring that quality assurance tests are regularly carried out and performance measurements made;
iii. Ensuring security, technical quality and technical performance of the digital asset.

- This committee shall meet whenever a technical decision about the digital asset is required, but not less than once per month (for performance review.)

6.4.4 Domain Management

MCDAs shall ensure internet domains are administered to ensure consistency with the dignity and high quality of the Government of Kenya

6.4.5 Domains

a) Name

MCDAs domain names shall be registered using the .gov.ke / .go.ke. Domain Providers reserve the right to waive this rule if the stated purpose is multi-jurisdictional in nature.

b) Relevance

MCDAs domain names must bear a direct semantic connection to the stated purpose. Furthermore, such names should represent a readily recognized concept associated with the stated purpose.

c) Format

Domain names must not:

1. be a personal name;
2. exceed 40 characters in length, including the first, second and top levels of the domain name
3. consist entirely of numerals;
4. have the same name as an already registered entity;
5. express a political statement or bear any semantic connection to a registered political party;
6. contain obscene or offensive language or otherwise prejudice the reputation or credibility of the gov.ke domain;
7. Infringe the intellectual property rights of other parties. It is the responsibility of the Registrant to ensure compliance with this requirement.
8. Have domain names that comprise common words should conform to the correct English/ Kiswahili spelling, grammar and syntax.

However, use of aliases is not forbidden.

d) Technical Composition

The domain must;

a) be at least 2 characters long;
b) short and simple
c) easy to say
d) easy to memorize
e) easy to spell and type
f) stable i.e. no need to change if the structure of the MCDA's changes
g) Contain only letters (a-z), numbers (0-9) and hyphens (-), or a combination of these but cannot start or end with a hyphen. Only one hyphen is allowed, start and end with a number or a letter, not a hyphen.

6.4.6 Web Design, Interoperability, Accessibility, Usability

MCDAs shall ensure websites are designed with consistent layout, usability and inter-operability

6.4.7 Design

a) All government websites shall be developed such that all the web pages are viewable in standard compatible web browsers, various operating systems such as Windows, Macintosh and Linux and devices such as PC, PDA, tablets, digital TV's and mobile phone based on the latest web standards.
b) Server-side scripting languages should be preferred over client side since client side scripting may face issues of browser compatibility, scripts being turned off by browsers, security, among others.
c) MCDA websites shall endeavor to use Cascading Style Sheets (CSS) as much as possible to control layouts/styles.

6.4.8 Interoperability

a) During web development, MCDAs websites shall validate to following (or later) technologies for published grammars:

HTML 4.01
XHTML 1.0
XML 1.0

b) Ensuring that Web pages and Web feeds are encoded in UTF-8 (UCS Transformation Format 8) character code

6.4.9 Accessibility

MCDAs shall ensure

a) Accessibility of web content allow enlargement or adapted to meet the needs and preference of different people.
b) Use of different user agents (software to access web content) are supported including desktop graphical browsers, voice browsers, mobile phone browsers, multimedia players, plug-ins and some assistive technologies.
c) Authoring tools used to produce web content

6.4.10 Usability

MCDA shall design website that is effective, efficient and satisfying by taking into consideration the user experience design.

6.4.11 Web Branding

MCDAs shall ensure that websites and portals display in a manner that is consistent with the dignity and authority of the Government of Kenya and which is attractive and Government-branded so that it is easily recognizable and usable by citizens

6.4.12 Main Element of the Page

a) Placement and precise measurement of the main elements of the page such as the banner, coat of arms, and primary menu must be precise and conform to the official graphic design template;
b) The mandatory content of the landing page of a public website is defined within the official graphic design template. This content must be prominent and should be refreshed on a regular basis.
c) The main landing page of a Ministry, MCDA or Department that incorporates a number of state departments may adopt an integrated format with an index to link to the landing page of each state department.
d) The landing page should have an index that links the page to specialized pages which have been organized to provide for the specific needs of citizens (G4C), Businesses (G4B) and Government cross-MCDA interaction where necessary (G2G).
e) Micro sites and Intranets should follow the same branding guidelines.

6.4.13 Fonts

A public website must be formal, clear and readable. Sans serif fonts with clean lines are preferred. It is preferable that only one font is used throughout but not more than three fonts should ever be used. Some suggested fonts for use on public websites; - Calibri, Arial, Franklin Gothic Book, Geneva, Trebuchet MS, Lucida Sans Unicode, Palatino Linotype, Times New Roman and any other good web font.)

6.4.14 Links and Pointers

a) Public websites should only link to approved public digital assets and approved social media. Links to commercial sites, commercial events and questionable external digital assets are prohibited. Generally, a link should be associated with a specific keyword or an approved graphic element (such as an official logo or an iconic photograph.) The mouse pointer should change in a standard fashion to indicate proximity to a link.

b) Active and read links should be marked through a standard colour-coding convention.

6.4.15 Social Media

a) The correct branded logo or handle for the specific social medium must be used at all times.
b) A link to a social platform should be for the purpose of accessing the associated page belonging to the public organ and not for commercial promotion of the social medium.
c) The public website of a public organ should only link to its associated social platform if it regularly ensures that content, breaking news and feedback are regularly updated on the social platform.

6.4.16 Multi media

Photographs are useful for achieving an eye-catching and attractive website. However, poor quality or busy photography can mar a website and slow down its loading speed. Therefore, MCDAs shall;

a) Use sharply-focused photographs of reasonable pixel strength;
b) Use tools to optimize their file size to its lowest possible value;
c) Caption them with a description of their date, location, event, persons present and photographer;
d) Consider the page proportions when choosing and placing photographs – they should enhance the text – not dominate it;
e) Do not stretch them in a direction that distorts the proportions of the image; and,
f) Since each photograph will download individually, keep the number of photographs on a web page to a minimum.
g) To retain the interest of the user, a web page should take between 3 and 18 seconds to load. Ensure that multimedia page elements do not slow down the loading of a page beyond this point.
h) Avoid automatic streaming of Voice, video and Podcast that can unwittingly use up a user's credit when they are using a mobile phone to browse your website.

6.4.17 Online Visibility

MCDAs shall use search engine optimization (SEO) techniques to ensure that the website is ranked highly by search engines and appears first on search lists for relevant web searches.

6.4.18 Legal Matters

MCDA shall take into consideration all legal matters from security, privacy, intellectual property, disclaimers etc.

i. Security

a) MCDAs shall host website in a secure location that conforms to Government of Kenya standards to protect the Government from the embarrassment of a defaced public website.
b) If hosting a web-based application which accesses public data MCDA must ensure that the database is isolated from the website and fully secured.
c) MCDAs must make every effort to ensure that the website does not harbor any malicious code and that it is not being used to harm users.

ii. Privacy

a) Any information about users that is collected through MCDA website must be secured and protected against unauthorized access.
b) The text and multimedia published on your site must have the necessary permissions for publication and should not violate the privacy of the subjects.
c) MCDA shall always inform a user when and why you are using a cookie to gather information about her.
d) User information gathered by a cookie shall only be used to improve the users experience on the website and should be deleted as soon as it is no longer necessary.

iii. Intellectual Property

a) The copyright of all content displayed on a website should be visually acknowledged.
b) All content that is not the copyright of the Government of Kenya should only be displayed after the necessary permissions or licenses are obtained from the copyright holder.
c) The source of all content items should be acknowledged through a properly formatted citation.
d) The conditions for use of Government of Kenya copyright materials displayed on your website should be stated at a suitable location on your website.

iv. Disclaimer

MCDA shall;

a) Always display a legal statement stating the boundaries of Government responsibility for content on the websites.
b) Inform the public that information on the site is placed in good faith and is general in nature and that before they commit to the information they must counter check to avoid any harm or losses.

c) Users must be warned of levels of accuracy of material facts. There should be a statement indemnifying the Government from the content on linked websites.
d) Keep a record of information on the site by backing up the site on regular basis.

6.4.19 Web Content

MCDAs shall develop content in a way that will keep stakeholders interested in the site.

6.4.20 General Considerations

a) Every image, video file, audio file, plug-in, etc. shall have an alt tag
b) Complex graphics shall be accompanied by detailed text descriptions
c) The alt descriptions shall describe the purpose of the objects
d) If an image is also used as a link, make sure the alt tag describes the graphic and the link destination
e) Decorative graphics with no other function shall have empty alt descriptions (alt= "")
f) MCDAs shall add captions to videos
g) MCDAs shall add audio descriptions
h) MCDAs shall create text transcript
i) MCDAs shall create a link to the video rather than embedding it into web pages
j) MCDAs shall add a link to the media player download
k) MCDAs shall add an additional link to the text transcript
l) The page shall provide alternative links to the Image Map
m) The ←area→ tags must contain an alt attribute
n) Data tables shall have the column and row headers appropriately identified (using the ←th→ tag)
o) Tables used strictly for layout purposes do NOT have header rows or columns
p) Table cells are associated with the appropriate headers (e.g. with the id, headers, scope and/or axis HTML attributes)
q) MCDAs shall make sure the page does not contain repeatedly flashing images
r) MCDAs shall check to make sure the page does not contain a strobe effect
s) A link shall be provided to a disability-accessible page where the plug-in can be downloaded
t) All Java applets, scripts and plug-ins (including Acrobat PDF files and PowerPoint files, etc.) and the content within them are accessible to assistive technologies, or else an alternative means of accessing equivalent content shall be provided
u) When form controls are text input fields use the LABEL element
v) When text is not available MCDA shall use the title attribute
w) MCDAs shall include any special instructions within field labels
x) Make sure that form fields are in a logical tab order
y) Include a 'Skip Navigation' button to help those using text readers

6.4.21 Web Hosting

MCDAs shall host websites securely and ensure the websites are updated to mitigate any cyber security threat.

6.4.22 General Considerations

a) MCDAs shall ensure that websites are hosted in a secure location that conforms to Government of Kenya standards to protect the Government from the embarrassment of a defaced public website.
b) If MCDA are hosting a web-based application which accesses public data, MCDAs shall ensure that the database is isolated from the website and fully secured.
c) MCDA must make every effort to ensure that websites do not harbor any malicious code and that it is not being used to harm the users.
d) The web master shall ensure that they are using the latest/updated CMS and ensure that the CMS are regularly patched.
e) Website should be regularly scanned for vulnerabilities and action taken in any.
f) The web master/web management committee should track how many users have the key/ root password and adhere to other security requirement for applications.

MCDAs shall always liaise with the host to ensure security of the websites.

g) Web master shall always ensure root/critical files are secured from unauthorized disclosure.
h) MCDA shall always adhere with standards on data security, and information management and other security requirements while hosting website and other online systems.

6.4.23 Monitoring and Evaluation

a) MCDAs shall monitor and evaluate websites to ensure their availability.
b) MCDAs may conduct customer survey to ensure that they meet the customer needs.

7.0 INTEROPERABILITY

There needs to be an agency-spanning co-operation by linking together existing special applications and using cost-intensive software components. To achieve interoperability in e-government systems;

7.1 MCDAs should identify and document their data needs

7.2 MCDAs should categorize their data elements in terms of primary (Must Have) data needs and secondary (Could have) data needs.

7.3 MCDAs should consider loosely coupled systems as opposed to tight coupled systems so as to allow for data interchange with other e-government systems.

7.4 MCDAs should ensure acquired / developed systems can accept and use data from other government agency systems

7.5 Adoption of XML Technologies should be preferred for integration of information-system and presentation of data.

7.6 Each MCDA should identify the list of processes belonging to it for standardization. The standardized processes should be made as e-Processes or e-Services; these should be shared with other stakeholders by the respective public agency through a Process Agreement. List of information on input, output and internal operations of each process/service should be shared with others.

7.7 Component-based e-Services model should be created by MCDAs so that existing components can be reused as much as possible.

7.8 The technology-neutral or technology-independent frameworks/formats should be used to ensure long term preservation of information.

7.9 The representation of information should be based on open standard and formats. The mandatory adoption of notified standards will help MCDAs to avoid vendor lock-in.

7.10 Solutions (proprietary/open source) based on Open Standards should be preferred.

7.11 While identifying stakeholders for various mechanisms, the following roles & responsibilities should be clearly defined and agreed upon:

- Ownership rights
- Assurance about correctness and integrity
- Assurance about making the data available for sharing within stipulated time period
- Addition of data in centralize repository
- Updating, auditing, compliance review rights and corresponding stipulated processes and discharge legal obligations etc.

7.12 Each MCDA should determine access restrictions within its own information system.

For the detailed description of the e-GIF that support the SGP refer to Annex 7

8.0 SYSTEMS INTEGRATION

PURPOSE

System integration requirements form a critical pillar of the Systems and Applications Standard. These requirements are mandatory and must be adhered to by all members of project teams involved in development and implementation of enterprise applications, including employees, consultants and / or contractors involved in the development or modification of integrations that support the MCDA at an enterprise level.

The scope of this standard extends to and includes integrations with in-house enterprise applications as well as commercial off the shelf enterprise systems.
In order to enable simple integration of existing or new systems, the system in use must include well-defined and well-documented interfaces. The openness of an e-government application is one of the crucial factors for its successful use.

The detailed standard is attached in Annex 8.

9.0 SOFTWARE LICENSE MANAGEMENT AND USAGE GUIDELINES

The intention of this Software Licensing and Usage Standard is to ensure appropriate usage of software and full compliance with existing software agreements, by all users across public service. The goal is to embrace all best practices for software usage and allow enough flexibility to serve the needs of public officers and citizens at large.

9.1 Baseline Inventory:

A major component of the Software Licensing and Usage Guidelines is data collection. The data will be used for reporting, monitoring, and decision support. The baseline inventory is a database with information about every computer in use within an organization (MCDA), as well as all of the software installed on these machines. Inventory information will be updated regularly by the ICT Head responsible. The inventory database should contain sufficient data to track the location of the computers, software usage history, maintenance activities, release and patch update history, and other relevant system-related information.

9.2 Purchase Information:

IT department through the ICT Head will keep proof-of-purchase documentation for all software purchased through the department.

9.3 License Certificates:

In most cases, software vendors provide license entitlement certificates once software is purchased. IT department will keep these on file to and produce them for verification on demand.

9.4 Software Profile Documentation:

IT department will maintain a profile for each software package in use. The profile will contain contact information for the software vendor, a link to the vendor support website, a record of the registered username and password (if applicable), description of the licensing information, a schedule for updates, maintenance and renewals, as well as serial numbers/keys. Specific arrangements will be documented for the utilization of the software, e.g. enterprise agreement, software metering, open source, etc.

9.5 Compliance:

Compliance will be an ongoing process and involves information collection and coordination with software vendors. The information collection will allow IT to adequately analyze current and future needs in terms of software replacement and new acquisitions. IT department will proactively work with software vendors to determine the most cost effective licensing arrangement based on projected usage of the software.

9.5.1 Compliance Documentation:

In order to be fully compliant with the software agreements;

a) IT will maintain proof of purchase of the license, the license agreement, the software certificate (if applicable), and an accurate record of individual licenses given out to various users, if the agreement is not a site or enterprise license. IT will maintain these records in a database to and enforce compliance at all times.

b) IT will review existing software agreements periodically. If individual licenses are close to being exhausted for any particular agreement that is not a site/enterprise license, IT will research the possibility of negotiating with the software vendor and purchasing more licenses, if deemed necessary.

9.5.2 If it is discovered that the MCDA is out of compliance with an existing software agreement, IT will take measures to ensure immediate compliance. This could include discovering that software has been installed on more machines than anticipated. If this is the case, IT will either

a) Remove the software from any extra machines, or

b) Negotiate with the software vendor for more licenses. If the latter, the department(s) using the extra licenses may be asked to contribute to the expense of purchasing more licenses.

9.6 Program Support

9.6.1 Reports:

IT department relies on the accuracy of reports in order to make sound decisions on software agreements and renewals. The reports include information on software, licenses and utilization, and should be included in the organization's asset register. The expected reports include:

a) Software License Compliance – For software with license key seats greater than 0, report the current deployed seats.
b) Computer Inventory Detail – Detailed listing of all computers on the network with full details about the installed software.
c) Software Inventory by Vendor – Software inventory listing grouped by vendor showing number of seats deployed.
d) Software Title and Version – Computer list: This report lists the computers having each software title in inventory.
e) Software Titled Deployed Count – Software inventory listing sorted by software title showing number of seats deployed.

9.6.2 Communication:

The goal of IT department should be to increase awareness among staff and users about appropriate use of software. Specific information will be provided to all users so that they understand what IT is tracking and how the information is reported. Communication will be done in a variety of formats including email distribution, and in some cases, facilitated presentations.

9.6.3 Software Distribution:

Any IT-funded software which can be distributed to staff, under a licensing agreement with a vendor, will be documented by IT. In order for a request or to receive the software, they will need to complete a formal request form. This will trigger the assigning of a license to the requestor (and thus adding a record to the license tracker) and a ticket will be created to have the software installed.

9.6.4 Home Use Software:

There is currently no Home Use Rights for Government licensed software which i.e. personal devices are ineligible. If you leave the public service, you will be required to hand over the laptop or desktop issued before your employment is terminated.

9.7 Licensing Arrangements and Definitions:

9.7.1 Concurrent use licenses require the purchase of a sufficient number of licenses based on estimates of how many computers are actually using the software at any one time. For example, if IT only has 20 concurrent use licenses for a particular software program, no more than 20 users can access that software at the same time.

9.7.2 Enterprise (site) licenses require a set fee to be paid, either a flat rate unrelated to the number of computers or, more commonly, on a scale based on the number of computers or other devices attached to the network, or on the number of staff.

9.7.3 Workstation licenses require the purchase of an individual license for every computer on which the software is made available. These are in limited quantity.

9.7.4 Vendors provide different licensing schemes at different costs; it should be ensured that the licensing scheme chosen by the MCDA satisfies the organization needs; however MCDA must not purchase a licensing scheme which is of a higher price than another licensing scheme provided by the same vendor that satisfies the MCDAs' needs. Documentation detailing how MCDA arrived at a chosen licensing scheme should be retained and availed when needed.

10.0 SYSTEMS GOVERNANCE STANDARD

The purpose of this document is to outline the systems development acceptable procedures which form part of the Systems and Applications Standard.
Effective development processes are critical to the success of systems development projects. This Systems Development Governance Standard provides:

a) Development standards for all stages of the System Development Life Cycle
b) Minimum requirements for software development activities, deliverables and acceptance sign-off.

These requirements are mandatory and must be adhered to by all members of the development teams, consultants and / or contractors involved in the development or modification of mission critical applications that support the MCDA at an enterprise level. The detailed standards are in Annex 9

ANNEXES

Annex 1 Enterprise viewpoint: fundamentals of e-government		
Frame of reference for e-government	**Organizational requirements**	The cross-administration approach Certain organizational requirements must be fulfilled in order to ensure the sustainable introduction of e-government. The most important of these requirements are described in the following sections. What is generally needed is co-operation, networking and co-ordination within and between administrative level. As far as administrative procedures are concerned, insular and go-it-alone solutions must be avoided when new applications are introduced. New solutions must be co- ordinated between the different levels in order to achieve maximum service depth and width on the largest national scale possible and in order to ensure the compatibility of administrative levels. In order to enable the nation-wide approach cabinet issued a circular of October 2015 where all agencies are required to work with ICTA on issues of e-government applications to avoid duplication and encourage reusability. At the same time, e-government co-ordination is to be improved and the transfer of solutions is to be speeded up. This avoids parallel development, saves costs and integrates, modernizes and optimizes administrative processes.

	Process optimization	The successful introduction and implementation of e-governments calls for preparatory restructuring activities on a process level. Existing rules, processes and structures must be adapted and improved because electronic forms of rendering services would otherwise stumble into the same fundamental problems which are also en- countered in conventional workflows not based on information technology. Existing administrative processes are partly the result of historical developments and have become extremely complex during the course of years as a result of many small changes. The following measures are hence recommended before special and technical applications are implemented. a. Simplification of processes and procedures b. Deregulation c. Shortening of process chains d. Reducing interfaces e. Avoiding iteration f. Reducing cycle and dead times24 This initiative is determined to achieve the fastest possible simplification of processes and statutory provisions concerning frequently used services involving multiple administrative levels.
	Qualification of personnel	The use and updating of standards means a continuous exchange of information and training process. Training people in the use of a PC costs more than the PCs them- selves, but also yields a more sustainable effect. Public service staff were found to be highly motivated to support e-government. This important asset must be exploited and increased in the interest of implementing e-government. Focal issues include intensive staff training as well as increasing the attractiveness of jobs in public administrations for IT experts.
		The use of e-government is strongly dependent on customer acceptance of the services offered. Full utilization of the savings potential of e-government is contingent upon the online services provided being accepted and used by potential users.
	Involvement of users	Expectations among citizens, companies and public agencies as the specific target groups need to be identified on an ongoing basis.
		The service portfolio and the service rendering process must be adapted to these expectations.
Legal frame of reference	**Electronic signatures**	Legal guidelines must be considered in addition to the organizational frame of reference.

		The legally binding nature of electronic communications is a crucial success factor for the implementation of e-government. What is hence needed is a digital solution for a signature with legally binding effect, i.e. the electronic signature. The legal adjustments necessary to enable the use of electronic signatures in the various agencies need to be done.
	Data protection	E-government offers a host of options and rationalization potentials in the IT sector. Ideally, data from the most varied contexts is gathered once only by a central function and is subsequently available to any de-centralized purposes and uses.
		However, when electronic data is exchanged within and between public agencies, data protection requirements must be considered and implemented by way of suit-able technical and organizational measures. Personal data, in particular, may not be gathered, processed or disclosed for any purpose other than the use explicitly contemplated by law.
	Barrier-freedom	People with disability (impaired vision and physical handicaps), depend on technical aids as a precondition for using the Internet. In order to op- timely enable these devices for e-government applications, a host of rules and requirements must be considered during programming, designing and editing. Agencies are required to ensure as much as possible that this group is well catered for in pursuant to the constitution on freedom to information and Equal Opportunities with the aim of overcoming and preventing disadvantages for disabled people in accessing the online services.

Frame of reference for e-government applications		
Interactions in e-government	Interaction levels	E-government services can be generally broken down according to interaction levels, I.e. information, communication and transaction. Information primarily covers the provision of information for the people, for businesses and other elements of society. Users on this level merely act as recipients of information. Many of these information systems are supplemented by communication solutions with dialogue and participation offerings which enable the exchange of news, messages and information. This offer ranges from simpler solutions, such as e-mail or web-based discussion forums, right through to more complex applications, such as video conference systems for tele-cooperation. Transaction applications represent the highest interaction level. This sector covers the real rendering of services by organization. These applications include, for example, the electronic receipt and processing of applications or orders as well as the provision of forms which can be filled online , Electronic payment etc. The electronic signature is an important element that ensures the authenticity and confidentiality of the data exchanged between the different parties.
	Interaction relations	Besides the classification in terms of interaction levels, the different partners involved in e-government can also be distinguished. a. Government to citizen (G2C) This situation refers to the electronic interaction between citizens and administrations. This area also covers non-profit and non-governmental organizations. b. Government to business (G2B) This term covers electronic relations between administrations and business. c. Government to government (G2G)

Transactions in e-government		As already mentioned, public administration services not only cover the field of pure services, but also rights and obligations. A functional classification of administrations is necessary as a precondition for standardizing the different types of administrative activity – and hence the possible transactions. Generally valid types of transactional services can be identified on this basis. **Transactional service types** The public administration can be divided into service and intervention functions based on responsibilities and legal forms. Different services types can be identified and classified as service-type and intervention-type services on the basis of the different categories of functional administrative branches. Services are demanded, i.e. initiated, by citizens or businesses from the administration. Services include: Sub-steps, actions and roles of transaction services The individual transaction types can be broken down further into individual sub-steps. Sub-steps consist of one or more actions in which different actors are involved. Examples of sub-steps, actions and roles related to the service area are discussed in the following. This methodological approach can then be used as a basis for developing similar models for any other transaction type.

		As a precondition for applying for a service, citizens must first be given the opportunity to obtain detailed information. The information step is followed by the submission of the application. The application is passed on to the public agency and from there to the officer in charge. Other organizational units or public agencies may have to be asked for comments or information. As already mentioned, processes may have to be optimized or reformed in this field. The examination of the case is followed by a decision. This decision, again, may have to be sent to other departments or officers for information. Finally the decision is communicated to the applicant. If the decision corresponds to the applicant's request, the case is closed and funds are disbursed, if applicable. In this case, permanent control of the application of funds must be possible. The procedure ends with archiving as the last sub-step. If the applicant does not agree to the decision, remedies in law are available in the form of a protest or legal proceedings, for example.
	Sub-steps, actions and roles of transaction services	The individual transaction types can be broken down further into individual sub-steps. Sub-steps consist of one or more actions in which different actors are involved. Examples of sub-steps, actions and roles related to the service area are discussed in the following. This methodological approach can then be used as a basis for developing similar models for any other transaction type. As a precondition for applying for a service, citizens must first be given the opportunity to obtain detailed information. The information step is followed by the submission of the application. The application is passed on to the public agency and from there to the officer in charge. This means that for the services the sub-steps need to be defined and show the interactions and contain further explanations.

Modules for the implementation of electronic procedures		The analysis of service types explained above and the related identification of sub- steps, actions and rules can be used as a basis for identifying functional modules which – given the required configuration possibilities – can be used ´to implement different procedures using information technology. The potential applications of these modules are dependent upon the quality of the process analysis and the chosen software architecture33. The following types of basic modules can be defined in conjunction with the above- described procedure. a. User interface The analysis of the different roles leads to the need to develop certain basic modules which enable functions for access to the e-government application. This includes a uniform user interface which is easily remembered, as user and role management functions as well as functions for authenticating users in the system. b. Action modules The actions identified are implemented in the form of application modules, with priorities being defined, for example, on the basis of their potential frequency of use in the implementation of the business logic. De-centralised and central modules can be distinguished here. c. Integration and infrastructure modules The definition of basic modules leads to the development of software or network- based components which standardise communication between the basic modules.

ANNEX 2: Information Viewpoint	
Schema repository	Schemas ensure uniform data grammar, semantics and layout which ensures interoperability and data exchanged between the e-government applications The MCDAs are to ensure that common schemas and identical definitions of elementary data types are used throughout the system cycle. The use of XML schemas is encouraged.

ANNEX 3: Computational viewpoint		
	Requirements and preconditions	The computational viewpoint is introduced in order to offer technical assistance when drafting e-government applications, with special emphasis being placed on re-usability and interoperability. One central aspect in this context is the integration of special and technical applications into existing and future e-government architectures and infrastructures.

	Administration-specific preconditions and frames of reference	Design decisions for the establishment of a software architecture for e-government applications must consider certain requirements and frames **Administration-wide services** The MCDAs are discouraged from implementing applications independently especially in cases where multiple public agencies are involved in the rendering of a service. The rendering of online services without media inconsistency is a central goal of ICTA and therefore Insular solutions make this almost impossible or lead to high costs when it comes to linking the various systems together. MCDAs are encouraged to partner up with all the relevant departments/Ministries to ensure that the one solution can cater for all so as to reduce on duplications. **Process optimization** Besides avoiding insular solutions and parallel development work, the reorganization of process chains is also recommended so as to simplify complex administrative procedures. Simplifying processes in special procedures enables substantial cost savings when it comes to implementing special applications. Furthermore, error-susceptibility as well as updating and upgrading costs can be reduced significantly. **Data protection requirements** Another central aspect in design decisions is to ensure adherence to data protection requirements. Despite all the advantages resulting from the central, non-redundant storage of data, measures must be taken to ensure adherence to all applicable laws when storing and processing personal data. The software architecture must hence include certain security systems in order to ward off manipulation of data and attacks by hackers.

	Interoperability and reusability	There needs to be an agency-spanning co-operation by linking together existing special applications and using cost-intensive software components, such as a payment platform or modules for supporting electronic signatures. Reusability of software components and interoperability of the individual applications and components are indispensable preconditions for taking up these challenges. The use of standardized and reusable processes within the framework of a uniform and standardized software architecture can help reduce costs in the long term. This standardized approach leads to uniform interfaces when it comes to drafting and implementing software projects. The basic modules must be integrated into software architecture as a precondition for their use in conjunction with the implementation of special applications.

	Basic requirements for a software architecture	Any system to be developed must fulfill a number of general requirements as detailed below: **Security** Confidentiality, authenticity and reproducibility as well as compliance with the government Data Protection Act and the relevant security standards must be ensured in the use of e-government applications. **Reusability** Reusability of an e-government application or of one of its components is one of the central requirements which is to be achieved by adhering to the architecture. Redundant development of applications for similar or identical services is thereby avoided, so that cost savings can be achieved in the long term. Furthermore, the use of tried-and-tested modules enhances the quality of the entire system. **Flexibility** Adjustment to new frames of reference as well as upgrades are easily possible and/or at a reasonable cost. E-government applications must be designed in such a manner that modifications of or amendments to an application – resulting, for example, from changes in legislation, process optimization or use by other public agencies – can be carried out in an effective manner and at a reasonable cost. **Openness** In order to enable simple integration of existing or new systems, the system in use must include well-defined and well-documented interfaces. The openness of an e-government application is one of the crucial factors for its successful use.

		Scalability Distribution of an e-government application or its individual components must be possible without any problems. This is the only way to ensure the ongoing use of an application in an efficient and performant manner as use increases. Especially in the case of an application which is centrally operated, the number of public agencies using it is not definite, so that its future, cost-effective scalability must be ensured when the number of public agencies and users increases. **Performance** A short response time of an application is vitally important in order to ensure its widespread acceptance among citizens and businesses. Complex transactions of- ten require processing large amounts of data. The successful use of an application is contingent upon the user-friendly and performance provision of data. **Availability** Access to e-government applications must be permanently ensured. A permanently available application signals reliability and trustworthiness, so that citizens and businesses become more and more willing to use the application and to sup- ply the – typically confidential – data necessary for the transaction represented by the application. **Error tolerance** The system must be capable of handling unforeseen and invalid system states. Errors or unforeseeable events may not lead to a crash or uncontrolled system behavior which the user is unable to understand. Faultless, transparent operation of an application is a vital prerequisite for the user's trust in complex transactions.
		Updating capability Operation and updating of e-government systems should be as simple and easy as possible. External experts who were not involved in the development of the system must be capable of ensuring efficient system maintenance and updating even without longer familiarization time.

Architecture decisions	General	The software architecture outlined here involves several fundamental design decisions. These are the mandatory use of object-orientated software development paradigms and a component-based software development approach on this basis.
	Component-based software development	Component-based software development enables the compiling of software from existing components and their reuse. This system is expected to yield several positive effects, such as: • faster development and provision of the application • lower costs • higher quality • less complex structure • flexible application systems and modern system architectures • However, the use of component-based software development not only has positive consequences. We recommend the use of software components due to project cycle times and the high share of similar and comparable applications. In order to develop robust, reusable components, clear-cut functional definitions of the components are necessary in order to generate maximum benefits by reducing parallel development efforts.
	Separation of presentation and business logic	Separating presentation and business logic offers a technical solution for the optimum support of multiple presentation channels, such as different browser types or mobile devices, such as personal digital assistants (PDAs). Besides this aspect, the separation of presentation and business logic significantly enhances the quality of code structure, thereby substantially improving updating and trouble-shooting capabilities, flexibility, reusability and reproducibility whilst at the same time lowering costs in the medium term. Furthermore, such a separation enables the potential distribution of an application to several servers, with one server being responsible for the presentation tier and another one for the business logic. This has a positive impact on operation with regard to security, upgrading capability and scalability aspects.

	Separation of business and data logic	The separation of business and data logic leads to applications which are independent of the database type. At the same time, functionality is not directly dependent on the database via abstraction and performance, for example, by caching.
	Four-tier architecture	The implementation of the above-stated aspects leads to a multi-layer architecture with four tiers. The implementation of a special application in tiers with the inclusion of components calls for a clear-cut assignment of components to a specific tier. This facilitates the classification of components and implies formal definitions of their functionalities. The individual tiers of the multi-tier architecture are the client tier, the presentation tier, the middle tier and the persistence tier / back-end. The client tier is where users and application software interact. The data processed by the presentation tier as well as the user interface are visualized. The presentation tier is responsible for presenting the application data (for ex- ample, as a website). The middle tier, also called the application tier, accommodates the most important components for implementing the application logic irrespective of their presentation. This is where the program sequence is controlled. The data from the persistence tier is processed accordingly and passed on to the presentation tier where user entries are validated or authorization is granted, for example. An optional part of this tier integrates central components, legacy or ERP systems, when necessary. External services can be given access via application interfaces to the application without having to use the presentation tier. The persistence tier is responsible for the storage of data objects. It abstracts from the database. The back-end as a collective term represents functionalities of the operating system and specific databases

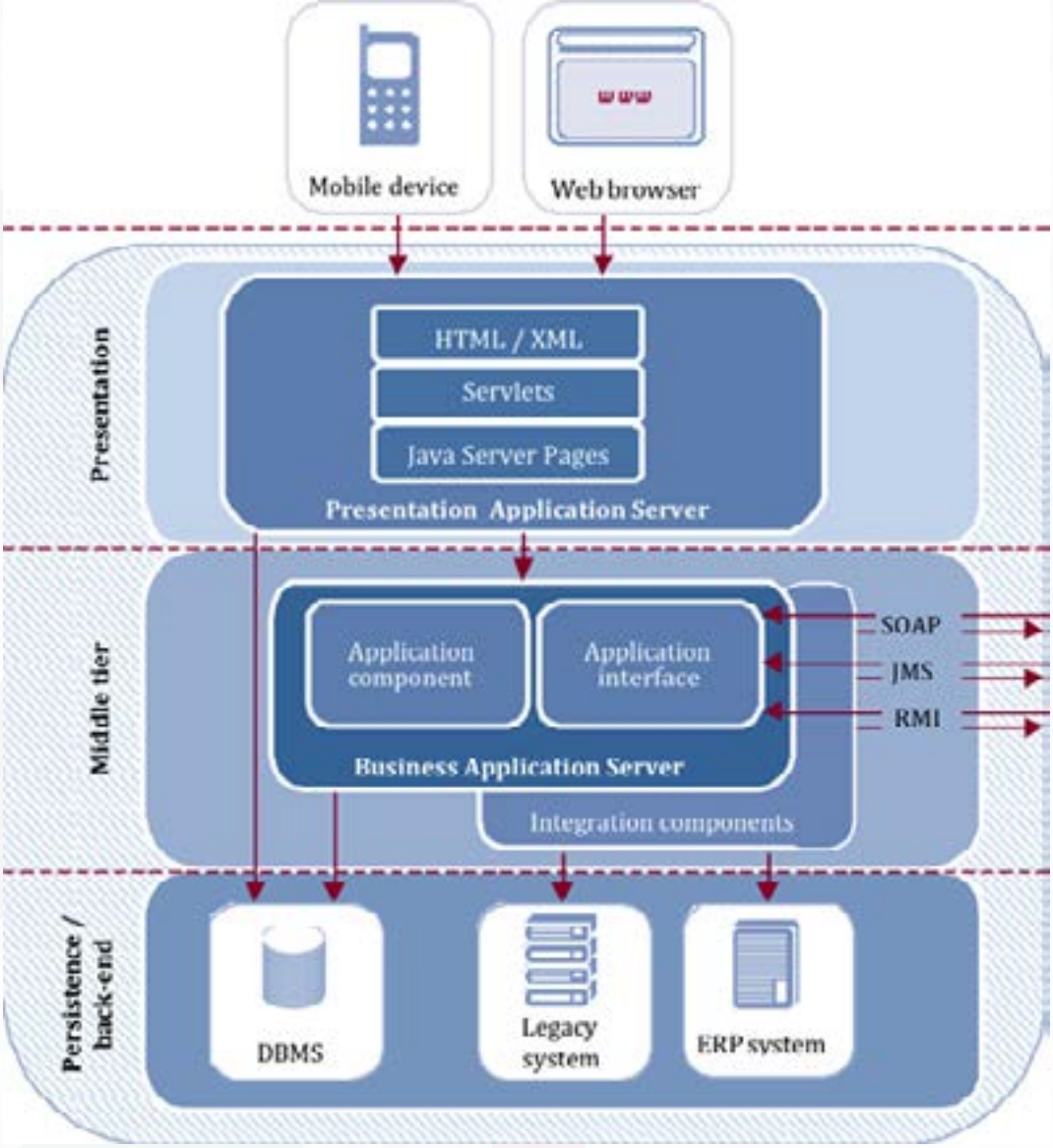


Figure A: four-tier architecture for e-government applications

The multi-tier architecture is preferably implemented using the Java programming language The decision in favor of Java is based on its platform-independence, optimum support of object-orientated software techniques, stability of the execution environment and the large number of free and commercially available APIs.

ANNEX 4: Engineering Viewpoint		
General		The selection of the appropriate infrastructure is a central success factor when it comes to planning, designing and operating e-government applications. A stable and secure IT infrastructure is the basic precondition for the reliable operation of e-government applications with high reliability. Today's data protection, data security, efficiency and availability requirements for e-government set high standards for operators of applications and infrastructures. The reference infrastructure for e-government applications is modeled on the basis of the engineering viewpoint according to RM-ODP and describes the encapsulation of system units and their connections. Not every public agency requires its own, complete e-government infrastructure. Smaller institutions may well use the Government Data Centre or sister agencies.
	Design of an e-government infrastructure	The introduction of a reference infrastructure in SAGA serves the aim of defining the infrastructural preconditions necessary for the operation of e-government applications and the required system architecture. The following goals are to be achieved by defining parameters or a reference infrastructure in the sense of an operating environment. a. Physical protection of systems b. Maximum availability of systems c. Increasing the security of systems and system components through classification on the basis of their protection demand d. Classification of systems and system components according to separate security zones e. Scalability of systems and infrastructures f. Simple service, efficient maintenance and updating of complex e-government applications and system components by operating personnel The figure below shows a general overall view of a distributed e-government application with the user, network and infrastructure areas.

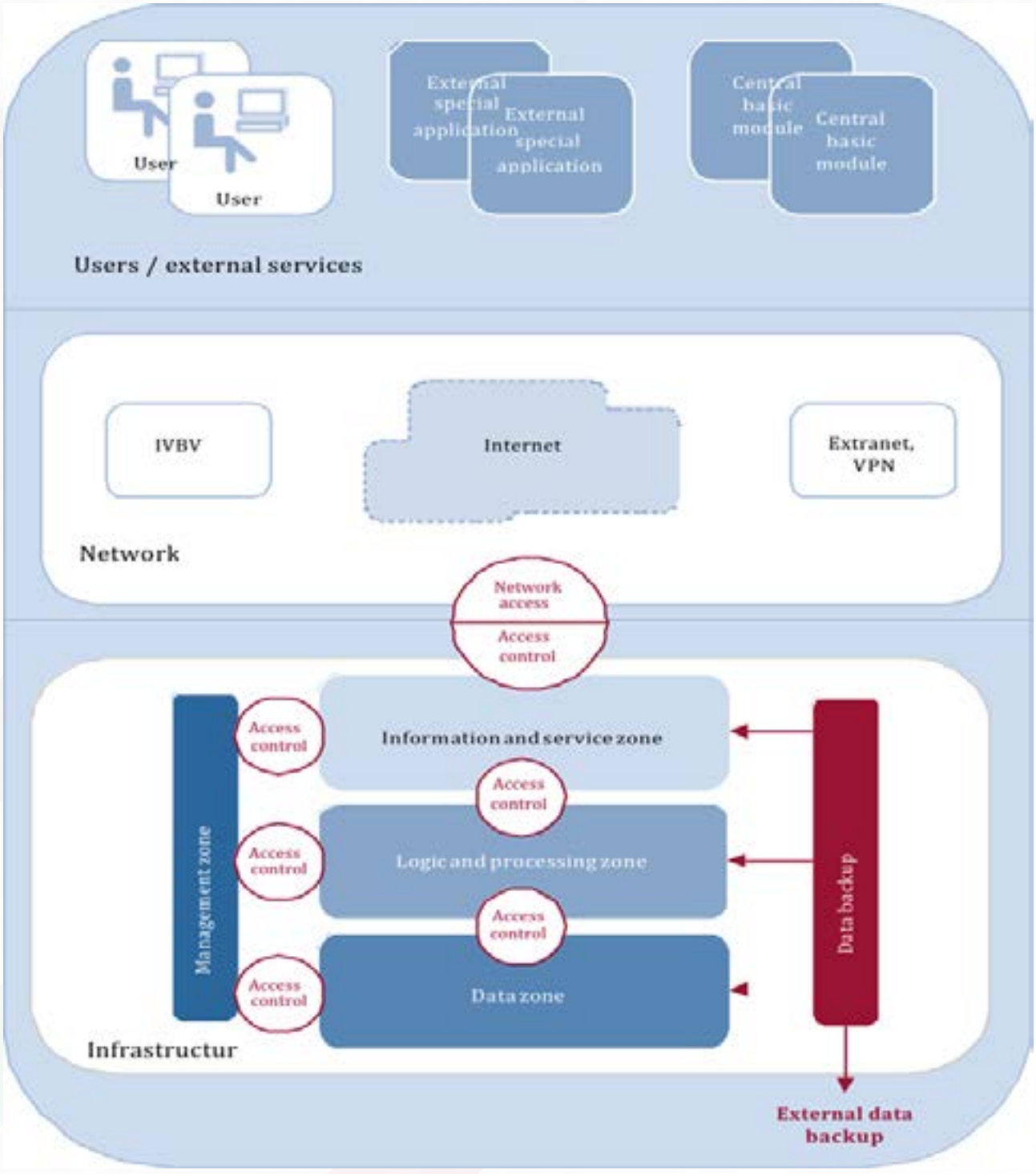


Figure 5

Engineering viewpoint of an e-government application

	Physical infrastructure	The protection of systems against external influences, the elements and unauthorized access requires the provision of suitable space. Computer centers designed to host e-government applications should hence at least feature the following proper- ties. a. Fire-resistant, structurally enclosed security space protected against radio interference b. Access control, including personal authentication c. Fire-extinguishing system with non-corrosive and non-toxic extinguishing agents d. Redundant power supply, including uninterruptible power supply e. Redundant air conditioning system f. Data backup media in a fire-resistant vault outside the computer centre
ANNEX 5 Technology viewpoint: Standards for the IT architecture		
	Process modeling	**Mandatory: Role models and flow charts** Role models and flow charts can be used to define simple processes. All the roles and systems related to a process must be identified, and the process steps must be described in the form of flow charts. Unified Modeling Language (UML) should be used for object-orientated model- ling for the preparation and documentation of large projects. Use cases are a particularly tried-and-tested way of creating and co-ordinating transparent specifications.

	Data modeling	**Mandatory: Extensible Markup Language (XML)** XML (Extensible Markup Language) is to serve as the universal and primary standard for the exchange of data between all the information systems relevant for administrative purposes. New systems to be installed should be capable of exchanging data using XML. Existing systems do not necessarily have to be XML-enabled. If necessary, it is also possible to use middleware which interprets incoming XML information and transforms or converts such information to the data format required by legacy and/or external systems. Mandatory: XML Schema Definition XML schemas according to World Wide Web Consortium (W3C) 55 are to be generated using the XML Schema Definition (XSD) for the structured description of data.
	Data transformation	**Recommended: Extensible Style-sheet Language** Transformation (XSLT) If applications use different XML schemas, conversion from one format to another can become necessary for data interchanging purposes. This format conversion is carried out via the XSLT56 language defined by W3C as part of XSL (Extensible Style sheet Language).

	Standards for data security	**Protection aims** Protection aims define the security interests of communication partners in a general form: **Confidentiality –** protection against disclosure to unauthorized parties: No data is made available or disclosed to unauthorized individuals, entities or processes. **Integrity –** protection against manipulation: Unauthorized modification or destruction of data is not possible. **Authenticity –** protection against faked identity/origin: Measures are taken to ensure that an entity or resource (such as an individual, process, system, document, and information) actually is what he, she or it claims to be. **Availability –** protection against failure of IT systems: The properties of an entity and/or resource can be accessed and/or used when this is attempted by an authorized entity. Information encryption (cryptography) is an important tool for securing confidentiality, integrity and authenticity. A high degree of availability is achieved through multiplicity, distribution and error tolerance.

ANNEX 6: PHASES OF SYSTEMS DEVELOPMENT

a. Purpose

The Project Proposal document is a clear statement of the problem to be addressed, and a brief description of the proposed solution. The document may be used to formally present notice that a development need exists. If there is a MCDA procedure for review of projects for the purpose of setting priorities, then the Project Proposal may be used for consideration of this project during this review.

If further work on the project is undertaken, the Project Proposal should serve as a base from which a full Requirements Definition (Traditional or Purchase Track) or System Definition (Prototyping Track) is developed. It may also be used as the basis for a Request for Information, if this phase is undertaken (Purchase Track).

b. Project Proposal Document

The Project Proposal document may be quite general. The document is usually prepared by the functional office. The following sections are ideally included in a proposal document:

- Problem. A brief discussion of the reason for the proposed new system or system modification should be given. Among questions to be addressed are: what benefits are to be derived? Why should the system be developed or modified?
- Description. A brief description of the proposed system or proposed modification to an existing system should be included in the Proposal. If the proposed system replaces an existing system, this should be noted.
- Users. The users of the new system or the system modification should be identified.
- Special Considerations. Any special requirements or constraints should be stated. For example, if National regulations or MCDA policy requires implementation by a certain date, this should be stated.
- Funding: The intended source of funding for both one-time development and acquisition costs and any on-going costs (for staff to maintain the system, and maintenance and/or upgrades of software and hardware) should be indicated.
- Interfaces. If the system includes any external interfaces, such as MCDA-to-MCDA interface, Electronic Transfer of Funds (EFT) to banks, or an Electronic Data Interchange (EDI) interface with vendors, then this should be explained.

c. Internal Audit Notification

Internal Audit should be notified, either by forwarding a copy of the Project Proposal or by notification via email or other means, that a project is being proposed.

d. Review and Approval

If the Proposal was not written by the functional office, then the functional office must approve the Project Proposal document. Approval of the Project Proposal document by the functional office does not imply a commitment to proceed with the remaining phases of development for the project.

1.0 REQUEST FOR INFORMATION (PURCHASE TRACK ONLY)

a. Purpose

The purpose of issuing a Request for Information (RFI) early in the development process is to gather information regarding currently available technology and products. This phase is only applicable to projects where purchase of a vendor product is being considered (i.e., this phase is part of the Purchase Track; Even for those projects where purchase of a vendor product is likely, this phase is not required, but may prove useful. Specifically, gathering vendor information prior to performing the detailed analysis of the functional requirements (Requirements Definition phase) may help to:

- establish the overall scope of the project, depending in part on the scope of the products available;
- identify whether or not a vendor package solution is feasible;
- clarify whether or not the business processes must be reviewed and possibly revamped, and if so, whether it is expected that business processes must be tailored to fit a vendor package (once a package is selected), or that processes will be modified, which could result in a package solution no longer fitting the business practices implemented;
- allow participants to learn about newer technologies available, that may, if utilized, affect business processes; and
- Allow participants to learn about the approximate cost of newer technologies.

Note: If it is determined as the result of reviewing the responses to the Request for Information that the business processes must be reviewed and possibly significantly modified, then this review should be undertaken before the Requirements Definition phase

Note: A Request for Information document may also be issued if a vendor may be engaged to perform a feasibility study or custom development of an application. This section may be used for guidelines in these cases, although the exact content of the RFI document will differ somewhat. Some information regarding vendor products may be gathered without a formal Request for Information, by researching available products, and contacting the vendors for standard information and brochures. For some projects, this informal approach to gathering information will suffice, and the more formal Request for Information process may be skipped. Other sources in addition to the vendors themselves may also be explored for information regarding available packages, including other MCDAs.

b. Request for Information Document

The Request for Information (RFI) document may be based largely on the Project Proposal document. At this point in the process, and for the purposes of gathering information from vendors, it is not expected that a detailed analysis of the functional requirements have been performed, and the level of detail provided in the Project Proposal should be adequate.
In addition to the information contained in the Project Proposal, the following should also be included in an RFI:

- An overview of the organization, and the context within which this system will be implemented (often an internal Project Proposal document assumes an understanding of the organization and would not include the type of overview that an external vendor might need). This should also identify the functional office or offices that will be the primary users of the system.
- An overview of the technical infrastructure within which the system must fit, if this has been determined. For example, if the system must run on specific hardware that is already installed, then this must be stated. However, unnecessary restrictions should be avoided at this stage.
- A description of all interfaces this system has with other applications.
- Any assumptions that have been made regarding the project. For example, if the intended new application will be an enhancement to an already existing system, where the assumption is that the existing system will remain in production more- or-less unchanged, then this should be explained.
- A clear iteration of the type of information desired in the responses to the RFI, regarding both the vendor product itself (the functionality, technical requirements, etc.), and also regarding the vendor and the support provided by the vendor. The RFI may also include whether or not vendors will be expected to provide a demonstration of their product.

c Process for Issuing a Request for Information

A Request for Information (RFI) must be submitted to the Purchasing Office for distribution. The department requesting issuance of an RFI may provide Purchasing with a list of companies to which the RFI should be sent. The Purchasing Office may add more companies, and may also post the RFI publicly.
The deadlines for responses established in the RFI must allow vendors a reasonable time period in which to respond. While this will vary depending on the complexity of the request and the project itself, generally between 30 and 60 days is reasonable. The RFI should include the name and number of a person at UC responsible for answering any questions vendors may have regarding the RFI.

d. Review of Responses

The vendors' responses to the RFI must be reviewed by those responsible for the project's next phase: Requirements Definition. Normally there is a project team composed of the Project Leader (and possibly other staff from the administrative computing department) and representatives from the functional office(s). For high risk systems, internal audit may also be involved.

If demonstrations of vendor products are given, then the same team should be present. While the review of RFI responses need not be as formal as a review of responses to a Request for Proposal, the project team should agree in advance how the review should proceed. Once all responses have been reviewed, the team should discuss the results and how this may affect the next phases.
One risk of researching vendor products before performing a detailed analysis of the functional requirements is having a specific product in mind during the Requirements Definition phase. The participants must be aware of this risk, and avoid premature attachments to specific vendor products.

2.0 System Definition (Prototyping Track only)

The System Definition phase can be undertaken as soon as the Project Proposal is approved, or as the first phase of a project where there is no formal Project Proposal document. The System Definition phase is part of the Prototyping Track (vs. the Traditional or Purchase Track) for systems development.

e. Purpose

The System Definition phase is intended to define the proposed system in sufficient detail for prototyping to begin. The most important outcome from this phase is an understanding and agreement with the functional office or offices on the proposed systems solution. Other outputs from this phase include:

- The System Definition document, which must include a description of the functional needs that the system must satisfy, and a high-level design of the proposed solution; and
- Data element definitions.

The Systems Definition document and the data element definitions may be completed iteratively, overlapping with the Prototyping phase. Prototyping of the system may begin based on a less detailed version of the document and data elements, and the document, data element definitions, and prototyped system may be refined in parallel.

f. System Definition Document

The primary purpose of the System Definition document is to describe the system from both technical and functional perspectives so that both the functional office and the application developer can understand what it will look like and do. The document may consist mostly or entirely of text, or may incorporate alternative descriptive methods, including charts or output from modelling tools. The following sections may be included in this document:
Scope and Objectives. The scope of the proposed system and the overall objectives are described in this section:

- The section should include a statement of the general purpose of the proposed system or system modification. If a Project Proposal was written for the project, then the general purpose should coincide with what was stated in the Project Proposal. If the purpose or objectives of the project have changed since the Project Proposal, then this should be noted, and the revised purpose and objectives stated.

- Every functional office which will use the new system should be identified, and their use

of the system described (e.g., one office may have read-only access to data for reporting, whereas another office has responsibility for update).
Current System or Procedures. This section describes the system (if any) or procedures currently in place. This section may also highlight the differences between the current procedures and the proposed procedures.
System Architecture. A description of the system architecture should be included, i.e., standalone PC-based, client/server, web-based, etc. If remote access to data is required (e.g., from other UC sites), describe how this will be provided.

Database Design. Every database within the system should be described. In as much detail as is possible at this stage of development, the structure and data elements should be included. For relational databases, the tables and the data elements within each table should be listed, as well as the relationship between tables, and any required referential integrity. Where useful, a picture of the database(s) may be included. Key fields should be identified, and indexed fields, both unique and non-unique, should be identified.

Any sequential files should be described. For sequential files, the data elements should be listed, the record format and length given, key fields identified (and whether or not the key fields are unique), and the sort sequence specified.
The data structure/schema needs to be well defined and should conform to the given Government data standards to ensure interoperability.
Sources of Data. Input sources for the proposed system should be fully described. The method for input should be identified, e.g., online data entry by the functional office, interfaces with existing systems, etc. For each input source, the functional office owning the data source (and responsible for the accuracy of the data) should be identified. Any verification of input data should be described. The frequency of data input and the approximate volume of data should be identified.
Some possible input sources include: Online data entry

The screens for online data entry should be described in general, and the processing for these screens described. (Details of the exact design of the screen may be left to the Prototyping phase.)
Data edits to be performed online for each data element entered by the user should be described.
Error conditions (data verification, and errors when updating the database) may be described.
Interfaces to the system
Any interfaces to the system should be described, including record layout (if available) and method of transmission of the data. If an interface file is used for a batch update process, the database update logic should be described, including data editing procedures and error handling.
For interfaces from other MCDA sites or from external sources, any issues regarding the definition of the interface or testing of the interface should be described.
Sufficient time for external sites to develop or modify the interface and provide test files should be incorporated into the Project Plan.

Other Batch Data Inputs

Any other batch update processes (other than for interfaces to the system) should be described, including: the database update logic, data editing, and error handling. Examples of other batch data input include: data originally collected on hardcopy forms and subsequently entered as transactions; or data files created by one process within the application, and input to a subsequent process.

Processing Logic. The processing logic for the system should be described in general. The logic for updating the database, via either online data entry or batch update, should be described (see Sources of Data, above). Processing for outputs, whether reports, interface files from the system to other systems, or online inquiry, should be described (see Major Outputs, below). Any processing, not directly related to input or output of data, should also be described. For example, for a system that will perform a modelling function, the logic for the modelling, including what data is used from the database and what data the user may enter or modify, should be described. Other logic, such as calculations to be performed, or workflow logic required, should also be described.
Major Outputs. All major outputs from the system should be described. This includes the following:

Reports (online or hardcopy)

Any reports from the system should be described, in general. Whether or not they are available online or via batch processing should be included in the description.

Interfaces from the system

Any interface files required from the system to another system should be described. If the record layout is known (e.g., the record layout is determined by the other system), then this should be included. If the frequency of required interfaces is known, this should be included. The processing to produce this file should be described in general terms.

Online Inquiry

Any online inquiry to data within the system should be described, in general terms. The method for providing online inquiry, and the types of screens to be made available should be described.

Ad-Hoc Inquiry and Reporting

If the system will provide ad-hoc inquiry and reporting access for users, then the general method for providing this access should be described. For example, if PC- based ad-hoc reporting tools will be used, this should be noted. If a specific product is expected to be used, this should be noted.
System Control and Data Verification. This section should describe how the integrity of the data is maintained within the system. The methods for verification of data accuracy should be described. Any verification of data that will occur outside of programs should be described. For example, if the functional office is responsible for reviewing certain reports before a specific process can be performed, this needs to be explained. Another example is when input data (e.g., from another system) is assumed to be accurate.
Audit and Authorization Features. This section should identify any audit, control or security and authorization considerations required. Examples are:

- appropriate record retention schedules to be followed to meet any legal or other data retention requirements for further information);
- Audit trails to meet legal or audit requirements; and levels of authorization required to satisfy privacy standards and/or the requirements of the functional office for security protection of the data.

For example, this might include read-only access to the data for selected users, or access limited to certain data elements for selected users. Authorization of users will depend on the authentication of the use rid used for access into the application. With respect to privacy policies, it is particularly important that any data elements relating to personal information (individually identifiable data) be identified in order for appropriate security measures to be addressed in the system.

Privacy Consideration. The privacy requirements which should be addressed in the development of each system are contained in section. In development of the system definition, these privacy standards should be reviewed. Any system requirements necessary for ensuring adherence to privacy standards should be described fully in this section of the System Definition document.
Backup and Recovery. This section describes the backup and recovery requirements in general terms. For example, if up-to-the-minute recovery is required for an online system (vs., for example, recovery to the end of the prior day), this should be noted.
Hardware/Software. For any shared resources, such as shared servers or printers, the impact of the proposed system should be evaluated to ensure that adequate resources are available to handle the new system without adversely impacting existing systems.

Any significant upgrades or new hardware resources required for the new system should be identified as soon as possible. The following types of resources should be considered:

- Memory allocation requirements on any shared server; disk space.
- usage on shared server machines;
- Client workstation requirements (e.g., for PC/Windows clients, minimum RAM required,
- Disk space required, level of operating system software required, and application software
- required);
- Printer requirements, including estimates of pages printed, and determination as to whether or not existing printers will provide sufficient speed for the output expected; graphics printing capabilities etc.;
- Networking requirements, including: workstation networking capabilities required (cabling and software), requirements for a LAN or access to a file server, etc.; and
- Level of client application software and server application software.

Impact on Existing Functional Office Operations. If there will be major impacts on the existing operations of the functional office with the implementation of the proposed system, then these should be described in general terms. Some examples of the possible impact of a new or revised system include impact on personnel (elimination of jobs or creation of new tasks), or reallocation of resources, such as from a centralized model to a decentralized model.
Conversion. This section should outline any conversions required for the system. If this system replaces an existing system, then the general approach for converting the existing data to the new system should be described. If there are major one-time start-up efforts (e.g., one-time data entry of historical data), then this should be described.

g. Data Elements

Draft data element definitions should be written during the System Definition phase. The data elements should be categorized by the major data entities identified (i.e., for relational databases, the table(s) in which each data element belongs should be identified).

Although these data elements may be further refined during the Prototyping phase, the application developer cannot begin a useful version of a prototype until the data elements are reasonably well defined.
At a minimum, the data element definitions should include the following information:

- Data Element Name
- Type of data element (numeric or alphanumeric)
- Format of the data element (e.g., for numeric fields, the number of digits allowed, such as 3v2 (meaning a number of the format nnn.nn))

General description or definition of the data element

If the data element is a coded value, then the possible codes and the corresponding interpretation of each code
Programming name (i.e., the name of the field in the database)
Additional information, such as data editing requirements and source of the data element, may also be included in the data element definitions.

If there is an official review process for data elements by a data administration group at the MCDA, then this review process should be initiated during this phase.
Such a review process may include a review for policy implications, and a review for consistency and standardization of data across administrative applications at the MCDA or OP. Issues of consistency and standardization are particularly important for applications where data will be shared by multiple departments, such as data warehouse applications. The data elements and any required review processes should be finalized by the end of the Prototyping phase.

h. Review and Approval

The Project Leader is responsible for determining when the System Definition document may be distributed. When distributed, copies of the System Definition document should be given to:

- the functional office(s) identified as users of the system
- those within the administrative computing department who should review the document
- Internal Audit (for high risk system)
- others, for informational purposes

The functional office should agree that the System Definition document accurately reflects their functional requirements for the new system or system modification, and accurately describes the agreed-upon systems solution. If the administrative computing department took the lead on producing the document, then a formal acceptance from the functional office (or offices) may be needed to ensure their agreement.
For some projects, the Project Leader may elect to require review and approval of the System Definition document from a working group of end-users in offices outside of the functional office(s) comprising the primary users of the system. This may be appropriate, for example, in the case of a MCDA-wide financial system that will impact offices outside of the Accounting Office, which is perhaps the project sponsor or major user.

The System Definition document is also given to Internal Audit for review for high risk applications. (Copies of the System Definition documents for applications not considered high risk are also available to Internal Audit upon request.) Audit's review of the System Definition document should provide feedback regarding whether or not necessary audit requirements have been identified (this is not to imply that Internal Audit is responsible for certifying the adequacy of the design and controls). In addition, Internal Audit should identify the degree of involvement desired by their office for the remaining phases of development.

3.0 PROTOTYPING (PROTOTYPING TRACK ONLY)

i. Purpose

Prototyping is a method of designing applications through iterative refinement of an actual working model of the system, followed by the acceptance of the actual working model of the system by the functional office. It involves developing a working model of a system (or a portion of a system), demonstrating the model to the prospective users, and modifying the model based on the users' responses.
The Prototyping phase is part of the Prototyping Track for systems development tracks.)
The major outputs from the Prototyping phase include:

- A working version of the system;
- Specifications for interfaces with other MCDA sites or an external organization, if applicable; and
- Updated Data Element definitions

j. When to Use Prototyping

Prototyping can be applied to almost any system. However, since it is based upon a high degree of involvement by the functional office in the specification of system functions, it is most useful for developing system functions with which end users will interact directly, such as online systems or online portions of systems. Also, this method works best when a prototype may be developed in a relatively short period, and the time between iterations of the working model is not overly long. For this reason, the method is best used for small to medium scale systems, or small to medium scale portions of larger-scale systems.
The following system functions, if applicable, may be difficult to address via prototyping, and the

Project Leader may choose to document these functions in a more traditional design document:

- Any interfaces to or from other systems;
- Backup and recovery of data.

k. Programming

The programming standards and guidelines in effect for the MCDA for the language being used for systems development must be followed -- refer to local standards manuals for further information.

l. Testing Each Prototype

The application developer is responsible for determining what testing is required for each iteration of a prototype, before presenting the prototype to the functional office for review.
A test plan, itemizing the test conditions to be exercised during testing, may be developed.
The test plan can be an informal list for use by the application developer only, or may be a more formal list if functional offices will be involved in testing the prototype or test verification.
In general, testing should address the following:

- every function performed by the prototype should be tested;
- for event-driven portions of the system (e.g., online processing), every event should be tested;
- any error conditions should be tested; and
- "boundary" conditions should be tested (for example, if an online program is designed to handle up to ten entries on a screen, then a test of entering ten or eleven entries and a test of entering zero or one entry should be performed.)

Test results must be reviewed carefully. The review of test results may be completed by the application developer, by the Project Leader, by the functional office, or by a combination of these staff.

m. External Interface Specifications

If there is an interface between the new (or modified) system and either another MCDA site or an external organization (e.g., a health insurance carrier), then specifications for this interface must be written. If the interface is FROM the other site or organization, then the specifications must be sufficiently detailed to enable the other organization to develop the interface file processing. If the interface is TO the other site or organization, then the specifications must provide sufficient information for the other organization to effectively use the interface file in their application. At this point in the development process, any general issues regarding the interface should have been resolved.
Interface specifications must include the following:

Overview of the data required for each file specified, including a description of the data.
Frequency of data submission (i.e., monthly, annually, etc.), effective dates of the submissions, and due dates (e.g., 5th working day of the month).
A record layout for each file transmitted must be included.

A list of data elements, including the format for each data element, must be included.

Physical file characteristics, including the data set name, record length and record format

OR
The file name to be transmitted for files to be transferred via ftp, and the target system for the ftp transmission.

The procedure for notifying the receiving site when the file has been submitted.

n. Data Elements

An updated list of data elements and data element definitions must be completed.

o. Review and Approval

When the actual working model of the system is completed, it must be presented to the functional office for sign off. For high risk applications, the completed model must also be made available for review to Internal Audit.

4.0 REQUIREMENTS DEFINITION (TRADITIONAL OR PURCHASETRACKS)

The Requirements Definition phase can be undertaken as soon as the Project Proposal is approved, or as the first phase of a project where there is no formal Project Proposal document. The Requirements Definition phase is part of the Traditional Track or Purchase Track for systems development.

p. Purpose

The Requirements Definition phase of systems development must define the functional requirements of the new system or system modification. The requirements must provide sufficient details for subsequent phases: the design of the system, or selection of a vendor package. Regardless of the split of responsibilities between the functional office(s), the administrative computing department, and any outside vendor for the Requirements phase, the participation of the functional office(s) during this phase is critical. A system will not meet the needs of the functional office if staff from the functional office(s) are not heavily involved in defining the functional requirements at this stage of the project.

q. Requirements Definition Document

Simply stated, the requirements definition document describes what the system is expected to do. While technical issues are not necessarily addressed during the Requirements phase, if certain technical requirements are already known, they may be included in the Requirements Document. For example, if a new interface is required with an outside agency, and the specifications are determined by the outside agency, then the details of the new interface (e.g., record layout, and frequency of the interface) may be included. Normally, however, technical details should not be determined before the functional needs are clearly identified.

The Requirements Definition document should include the type of information described in the sections below. The document need not include the specific sections listed, as long as the information is included. Certain sections may not be needed for some projects (e.g. privacy may not be a consideration for a system if no individually- identifiable data is included). The document may consist mostly or entirely of text, or may incorporate alternative descriptive methods, including charts or output from modelling tools.
Scope and Objectives. The scope of the proposed system and the overall objectives are described in this section:

- The section should include a statement of the general purpose of the proposed system or system modification. If a Project Proposal was written for the project, then the general purpose should coincide with what was stated in the Project Proposal. If the purpose or objectives of the project have changed since the Project Proposal, then this should be noted, and the revised purpose and objectives stated.
- Every functional office which will use the new system must be identified, and their use of the system described (e.g., one office may have read-only access to data for reporting, whereas another office has responsibility for update).

Current System or Procedures. This section describes the system (if any) or procedures currently in place. This section may also highlight the differences between the current procedures and the proposed procedures.
Sources of Data. Input sources for the proposed system should be fully described. The method for input must be identified, e.g., online data entry by the functional office, interfaces with existing systems, etc. For each input source, the functional office owning the data source (and responsible for the accuracy of the data) must be identified. The method for verification of the accuracy and integrity of the data must be specified. It should be noted if data accuracy and integrity is assumed to be outside the scope of the system (e.g., where data from another system is assumed accurate). The frequency of data input and the approximate volume of data should be identified.

For interfaces from other MCDA sites or from external sources, any issues regarding the definition of the interface or testing of the interface should be described. Sufficient time for external sites to develop or modify the interface and provide test files must be incorporated into the Project Plan.
Processing. The proposed processing to be performed by the system must be described in general. The logic for updating the database, via either online data entry or batch update, must be described. Processing for outputs, whether reports, interface files FROM the system to other systems, or online inquiry, should be described. Any processing, not directly related to input or output of data, should also be described. For example, for a system that will perform a modelling function, the logic for the modelling, including what data is used from the database and what data the user may enter or modify, should be described. Other logic, such as calculations to be performed, or workflow logic required, should also be described.

Major Outputs. All major outputs of the system should be described. This includes reports, and required interfaces to other systems, either within MCDA or to outside agencies. If the format and frequency of required interfaces from the proposed system to other systems are known, then this information should be included.
For systems that are replacing existing systems, or for system modifications, any changes to existing reports or elimination of any existing reports should be noted.
Interfaces with Other Systems. Generally interfaces with other systems will be identified in earlier sections either as a source of data (where data comes from another system to the new system), or as output (where data is sent to another system). Any interfaces not already described in earlier sections must be identified.

Database. While it may be premature at this phase of the project to complete a detailed database design, the basic data entities required for the system must be identified, and the relationship between entities should be identified. For example, in personnel systems within the MCDA, an employee is an entity, and appointment is often another entity. There is a one-to-many relationship between employee and appointment, since MCDA employees may hold multiple appointments.

Draft data elements (see section below) should be grouped by the data entities identified.

Audit and Authorization Features. This section should identify any audit, control or security and authorization considerations required. Examples are:

- appropriate record retention schedules to be followed to meet any legal or other data retention requirements;
- audit trails to meet legal or audit requirements; and
- Levels of authorization required to satisfy privacy standards and/or the requirements of the functional office for security protection of the data. For example, this might include read-only access to the data for selected users, or access limited to certain data elements for selected users. Authorization of users will depend on the authentication of the use rid used for access into the application. With respect to privacy policies, it is particularly important that any data elements relating to personal information (individually identifiable data) be identified early in the development cycle in order for appropriate security measures to be addressed during system design.

Privacy Consideration. In development of system requirements, these privacy standards should be reviewed. Any system requirements necessary for ensuring adherence to privacy standards should be described fully in this section of the Requirements Definition document.
Backup and Recovery. This section describes the backup and recovery requirements from a functional standpoint. For example, if up-to-the-minute recovery is required for an online system (vs., for example, recovery to the end of the prior day), this should be noted.

Hardware/Software. If there are hardware and/or software requirements for the system that are known at this point of the development process, these must be noted. For example, if there is a minimum configuration for a PC for use with the system (e.g., amount of memory required and level of operating system required), then this should be described.
Impact on Existing Functional Office Operations. If there will be major impacts on the existing operations of the functional office with the implementation of the proposed system, then these should be described in general terms. Some examples of the possible impact of a new or revised system include impact on personnel (elimination of jobs or creation of new tasks), or reallocation of resources, such as from a centralized model to a decentralized model.

Conversion. This section should outline any conversions required for the system. If this system replaces an existing system, then the general approach for converting the existing data to the new system must be described. If there are major one-time start-up efforts (e.g. one-time data entry of historical data), then this must be described.

r. Data Elements

Draft data elements and definitions should be written during the Requirements Definition phase. The data elements should be categorized by the major data entities identified (see the earlier section on Database). These data elements will be further refined during the design phases. For projects where a vendor package solution will be pursued, the data elements and the definitions will depend largely on the package selected, and should be finalized during the Installation phase.

For systems that will be developed in-house, if there is an official review process for data elements by a data administration group at the MCDA, then this review process should be initiated during this phase. Such a review process may include a review for policy implications, and a review for consistency and standardization of data across administrative applications at the MCDA. Issues of consistency and standardization are particularly important for applications where data will be shared by multiple departments, such as data warehouse applications. The data elements and any required review processes should be finalized by the end of the design phases.

s. Review and Approval

The Project Leader is responsible for determining when the Requirements Document may be distributed. When distributed, copies of the Requirements Document should be given to:

- the functional office(s) identified as users of the system
- those within the administrative computing department who must review the document
- Internal Audit (for high risk systems)
- others, for informational purposes

The functional office should agree that the Requirements Definition document accurately reflects their functional requirements for the new system or system modification. If the administrative computing department took the lead on producing the document, then a formal acceptance from the functional office (or offices) may be needed to ensure their agreement.

If the Requirements Definition document was primarily or entirely prepared by the functional office and the administrative computing department will be involved in design and/or programming (or maintenance) phases, then the administrative computing department must review the document to ensure that the requirements are adequately defined and that the proposed project is consistent with the overall technical strategies of the MCDA.

For some projects, the Project Leader may elect to require review and approval of the Requirements Definition document from a working group of end-users in offices outside of the functional office(s) comprising the primary users of the system. This may be appropriate, for example, in the case of a MCDA-wide financial system that will impact offices outside of the Accounting Office, which is perhaps the project sponsor or major user.

The Requirements Definition document is also given to Internal Audit for review for high risk applications. (Copies of the Requirements Definition documents for applications not considered high risk are also available to Internal Audit upon request.) Audit's review of the Requirements Definition document should provide feedback regarding whether or not necessary audit requirements have been identified (this is not to imply that Internal Audit is responsible for certifying the adequacy of the design and controls). In addition, Internal Audit should identify the degree of involvement desired by their office for the remaining phases of development.

5.0 REQUEST FOR PROPOSAL (PURCHASE TRACK ONLY)

t. Purpose

The purpose of the Request for Proposal phase is to engage in a formal process for eliciting bids from vendors when the project involves a vendor package solution. This phase is part of the Purchase Track.

Note: A Request for Proposal document may also be issued if a vendor may be engaged to perform a feasibility study or custom development of an application. This section may be used for guidelines in these cases, although the exact content of the RFP document will differ somewhat.

u. Request for Proposal Document

The core of the Request for Proposal (RFP) document is describing the functional requirements for the system, which has been accomplished as part of the Requirements Definition phase. The Requirements Definition document forms the basis of the RFP document.
In addition to the functional requirements contained in the Requirements Definition document, the RFP should include the following:

General Information.

- an overview of the overall MCDA organization and a description of the functional offices that will be using the system, plus a description of the overall technical environment in which the system will operate and of the administrative computing organization (whether it is centralized or decentralized).

- a schedule of key events, both for the proposal process (deadline for proposals (vendors must be given a reasonable amount of time to prepare responses), schedule for MCDA's decision, etc.) and planned implementation date (subject to change at the discretion of MCDA)

- response terms and conditions, to provide information regarding the formal procedure surrounding the RFP process, including:

1. who to contact for clarification of the RFP;
2. date and location for delivery of proposal;
3. official approval process and notification of award; and

- some definitions for the purpose of legal protection of MCDA, such as:

1. the right for MCDA to amend or supplement the information in the RFP;
2. the right for MCDA to not make any award as a result of this RFP process;
3. assignment or subcontracting (e.g., will the vendor subcontract any of the work);and
4. any issues regarding news releases and confidentiality.

Technical Requirements.

Any restrictions regarding the hardware or software to be used for the vendor package must be listed. If the product must run on currently-installed hardware and operating system and work within the current network configuration, then this must be stated. If the database management software, application development and/or language software, and system software must match that already installed at the MCDA, then this must be stated. If a required environment is described and there are known upgrades or major changes in the future, then these must be outlined as well.
Whether or not MCDA has specific hardware or software restrictions, all responses to the proposal must include a complete description of the technical environment required for the product. In addition, the vendor response must state how long the package will be supported in the given environment, and what major changes to the required environment are in the foreseeable future.

Any restrictions regarding how security and authentication must be handled by the package must be explained in the RFP. For example, if it is a requirement that the package handle security using DCE, then this must be stated.
Whether or not MCDA has specific security and authentication restrictions, the vendor must supply information regarding how security and authorization are handled within the product, if this is a requirement for the application.
Support requirements, both during the installation process and on-going support, must be outlined in the RFP.

The vendor responses must outline in detail the level of support that will be provided (and the associated costs), including: the expected staffing level during the installation process; the amount of training to be provided during the installation process both for the functional office(s) and for internal technical staff; on-going support, both for functional office questions (on-site and by telephone), and for technical problem solving and on-going maintenance of the application. Vendor responses must state whether or not MCDA-specific customization of the package is possible, and whether the vendor undertakes responsibility for custom changes, and the on-going maintenance of the custom changes. How customization of the package, if allowed, would affect implementation of future upgrades to the package must also be stated in responses.

Vendor responses must include a description of how upgrades to the package are handled (both responsibility for installation and the cost of upgrades); whether or not the vendor will undertake future customization of the package at the request of MCDA and how such customization would affect future upgrades; and how the vendor determines what enhancements are made to the base package.

Other Restrictions or Vendor Response Requirements. If there are other specific restrictions known at this point related to the anticipated vendor relationship or other information required from the vendor, then these should be explained. Possible additional items include:
Information regarding the vendor, such as:

- number of similar applications implemented for clients;
- years the vendor has been providing the types of services or software outlined; financial status; and
- Client references.

Any requirements related to cost and/or payment schedule, e.g., that the payments will be phased with a specified percent held until after final completion.
Any insurance that the vendor is required to carry.

Proposal Format and Instructions. The RFP should outline the format of the responses. A possible outline for the standard format of responses follows:

- Proposed solution: Table of contents
- Description of system solution
- Application software requirements
- System-level software and technical requirements Hardware requirements
- Proposed implementation schedule and level of effort On-going support
- Exceptions to the RFP (i.e., identification of any areas that the vendor cannot address)
- Sample contracts

- Sample documentation
- Vendor information and qualification and background information Client list
- Cost Proposal: Cost information, including: cost of the package, installation, and on-going support

v. Process for Issuing a Request for Proposal

Once a decision is made to proceed with a Request for Proposal process, the MCDA Purchasing Office should be notified. The schedule for preparation and issuance of the RFP document must include time for the Purchasing Office to review the RFP, if needed, and to perform the official distribution of the RFP. The Purchasing Office should be consulted during the preparation of the RFP, for example, if changes to the schedule occur, or if questions arise regarding the process. Once the RFP document is completed (normally by the project team), it is forwarded to Purchasing together with a recommended list of companies to which it should be sent. Purchasing then reviews the RFP, and may include additional companies in the distribution list. Purchasing distributes the RFP, and also posts the RFP publicly.

w. Review of Responses

It is important to establish an evaluation framework within which the responses to the Request for Proposal can be reviewed. This framework must take into account the requirements for the system as defined in the RFP document. It is helpful to have a weighting mechanism, so that the requirements may be prioritized. The review process is summarized, together with any forms used for the evaluation and a final recommendation for a selection, in the Feasibility Study (see next section: Feasibility Study).

6.0 FEASIBILITY STUDY (REQUIRED FOR PURCHASETRACK)

x. Purpose

The feasibility phase is only necessary under circumstances where significantly different options are available for providing a systems solution to the functional needs identified in the System Definition phase (Prototyping Track) or the Requirements Definition phase (Traditional or Purchase Track). For example, a feasibility study would be required:

- when significantly different approaches by the administrative computing department in meeting the functional office's requirements are possible;

- when there is a choice of either in-house development or purchase of a vendor product ("make or buy");and
- When vendor development or purchase of a vendor product is considered. In this case, the feasibility study should be used to evaluate different vendor responses to Requests for Proposals (see previous section).

y. Feasibility Study Document

The Feasibility Study may be prepared by the functional office or by the administrative computing department, although both offices will be involved in its review. In some cases, particularly for MCDA-developed applications where different options are possible, the Feasibility Study may be incorporated into the Requirements Definition or System Definition document.

The contents of the document will vary somewhat depending on the alternatives available for a particular project. Generally, the following should be addressed in the study:

Overview of the Proposed System. A brief overview of the proposed system should be included in the Feasibility Study, although the System Definition or Requirements Definition document should be referenced for further details.
Description of Alternatives. A description of each of the alternatives, or vendor proposals, should be made, outlining in broad terms the advantages and disadvantages of each alternative.
Features Comparison. A more detailed comparison of how each of the various alternatives meets the requirements should be presented. Often it is necessary to assign a priority to each major feature, since some features are critical or mandatory, while others may be only "nice to have". Any option which does not meet the mandatory requirements will be eliminated from consideration.

The attached form, "Features Comparison", provides an example of how this comparison might be displayed. On this form (or something similar), each major feature of the system is listed under the "Feature" column, each feature is rated in priority as Mandatory, Desired or Optional, and each alternative is ranked as to how it satisfies the system feature. For example, for a Human Resources application, one major feature might be retention of historical data, which the functional office considers a required, or Mandatory feature. If there were three alternatives being considered, e.g., three different vendor solutions, each vendor's product would be rated, perhaps on a basis of 1 to 5 (where 1 indicates that the feature is inadequate or missing from the product, and 5 indicates that the product provides an excellent facility for the given feature). This type of comparison provides a method for quantifying, to some extent, how well each alternative identified meets the system requirements.
Cost Comparison. The costs of each alternative should be fully indicated. Included should be both one-time costs (e.g., software development or acquisition and hardware acquisition or upgrade) and on-going costs (e.g., for staff to maintain the system, and maintenance and/or upgrades of software and hardware).

Compatibility. The compatibility or consistency of each approach with other existing applications and hardware or management's intended hardware and software environment should be considered. For example, an Oracle version of a software package with desired features may be available, but if all other applications currently in place are Sybase-based, then the impact, including the cost of training for and supporting an Oracle software package, may overshadow the advantages of the software. The compatibility of the proposed solutions with current procedures and operations should also be addressed.
Evaluation of Vendor. When the evaluation of alternatives includes consideration of vendor proposals, the following additional types of issues should be addressed:

- Vendor Profile/History

1. How many applications similar to that proposed has the vendor completed?
2. How many years has the vendor been providing the types of service or software which is proposed?
3. How closely related to the vendor's primary installations/services is the proposed solution?
4. What is the financial status of the vendor? Financial statements can be requested to ascertain the financial stability of the vendor.

5. What is the level of satisfaction of current clients? Client lists can be obtained to gather this information.

- Vendor Support

If the vendor will continue to provide support after installation, the following issues should be addressed:

1. The level of support provided should be evaluated. Typical measures are the number of support staff in each geographical area, the expertise of the support staff etc.
2. The normal range of response time for requested service should be identified, if relevant for the proposed application.
3. The cost for the on-going vendor support and/or product maintenance should be identified.
4. The cost of new releases of vendor software as they become available should be identified.

Schedule. Differences in the implementation time frame of the various options should be noted.
Recommended Alternative. A recommendation should be made for which alternative should be selected, with reasons given to support the recommendation.

z. Review and Approval

Functional office agreement with the recommended alternative and approval to proceed is required before work can begin on the next phase of systems development.
If the Feasibility Study was primarily or entirely prepared by the functional office and the administrative computing department will be involved in design and/or programming and installation (or maintenance) phases, then the administrative computing department must review the document to ensure that the recommended alternative is consistent with the overall technical strategies of the MCDA.

FEATURES COMPARISON FORM

Feature	Mandatory	Priority Desirable	Optional	Alternative 1	Alternative 2	Alternative 3

2. VENDOR CONTRACT AND INSTALLATION PLAN (PURCHASE TRACKONLY)

z. Purpose

Once a selection of a vendor has been made, a contract must be negotiated and completed with the vendor. Once the contract is in place, an Installation Plan (basically, a Project Plan specifically for package installation) must be developed. These steps are only required when purchasing and installing a vendor package (i.e. these steps are part of the Purchase Track.

aa. Vendor Contract

The terms of a contract with a vendor of a systems package are extremely important, particularly if the relationship with the vendor will be on-going (e.g., the vendor will provide on-going support for the package after installation). An inadequate contract could result in unexpected costs to MCDA or problems with providing adequate support for the system, particularly in the event of difficulties arising in the relationship with the vendor.

The terms of the contract must clearly delineate the responsibilities of the vendor for the project. This must include one-time tasks, such as the following: customization or specialized programming; conversion processing; testing; installation of the package; training of both functional and technical staff; and any follow-up issues with respect to training or problem solving. If installation of the package will involve any modifications or additions to the technical environment (hardware, systems software, or networking capabilities), then the vendor's responsibilities in these areas must be listed. If the vendor will be involved in any on-going support, then these responsibilities must also be explained in the contract, including, for example: answering questions from the functional office on use of the package; technical problem resolution; further customization or specialized programming; installation of maintenance releases or upgrades to the product; and on-going maintenance of the product.
In addition to clearly explaining the vendor's responsibilities, the following issues must also be addressed in the contract:

- The costs and payment schedule must be included. It is recommended that the payment schedule be phased, and dependent on specified milestones or deliverables. It is also recommended that some portion of the total payment is held (e.g., 10%) until the installation is complete, to ensure the vendor's completion of the project. Often contracts will include a good faith deposit made by MCDA to the vendor at the beginning of the project.

- Requirements for the vendor to carry insurance must be specified in the contract. This would include, for example, Worker's Compensation insurance for their staff, liability insurance, and possibly insurance related to the product itself.

- The contract should specify how changes to the contract will be handled. In any major project, some changes must be expected, although vigilance should be exercised by both functional office and administrative computing staff to limit the number and the extent of changes that occur at this stage of the project.

However, the contract should protect MCDA as much as possible from excessive additional expenses being charged by the vendor in the case where some reasonable changes (albeit possibly outside the scope of the contract) are requested. This portion of the contact should specify an hourly rate for vendor staff to perform changes or additional work beyond that anticipated at the time of the contract negotiation.

bb. Development of Interfaces and Customization of the Package

In most cases, installation of a package will involve some programming. The two most likely categories is modifying or rewriting interfaces between the package and other installed applications, and any customization that will be done to the package prior to installation. Any programming required as a result of installing a package must follow the phases for one of the other two development tracks, Prototyping or Traditional Life Cycle, which include phases for analysis, design, programming and unit testing. For these phases, the work may be handled as a completely separate project (or multiple projects), with separate Project Plans. These projects must then converge prior to the System Testing and Implementation phases.

cc. Installation Plan

While an Installation Plan does not differ in concept from the Project Plan that exists for every project, the installation of a vendor package will often consist of a large number of tasks performed by external vendor staff, internal administrative computing staff and functional office staff, often under strict time frames, and requires careful oversight for successful completion. Certain aspects of the Installation Plan are specifically related to installation of a vendor product. As for any Project Plan, the Installation Plan primarily consists of a list of tasks required to complete installation of the package, together with who is responsible for performing each task and a deadline for completion of the task. Logical sets of tasks may be grouped, and completion of a set of related tasks is a project milestone.

In addition to containing the tasks required for testing and installation of the product, the Installation Plan must combine implementation dependencies and dates for all "projects" that may be related to the installation of the package. This would include any programming performed separately from the package installation for interfaces or customization of the package. All related projects must converge, prior to System Testing and the final Implementation. The Installation Plan is a key tool to help ensure that this convergence occurs smoothly. The Project Leader is responsible for coordinating across all related projects, and ensuring that work is progressing satisfactorily on all fronts.

dd. Coordination with Vendor Staff

Even when responsibilities of all participating groups (functional office or offices, administrative computing department, and vendor staff) are clearly defined and well understood, on-going coordination between all groups, and particularly with vendor staff, is critical to a successful installation. The Installation Plan is an important tool in ensuring that everyone understands the tasks and responsibilities, and that tasks are completed on time. In addition, other tools may be used to assist in the coordination process, including, for example:

- regular (e.g., weekly) meetings between representatives from each group, to review the Installation Plan and progress on the active tasks, and to resolve any issues that have arisen;

- regular conference calls, daily during periods of intense work, to ensure that work on the tasks is progressing, and that any problems are dealt with quickly; and
- An on-going list of any outstanding issues that arise, together with the person or group responsible for the resolution (or for the next action required for resolution).

This issues list differs from the task list on the Installation Plan, since generally outstanding issues require research or discussion before they are resolved, rather than well-defined tasks to be performed. Resolution of an issue may result in additional tasks being added to the Installation Plan.

2.0 GENERAL DESIGN (TRADITIONAL TRACKONLY)

a. Purpose

The General Design phase results in development of an overall system design based on the approved Requirements Definition document, and on the chosen approach selected in the Feasibility Study, when present. The General Design expands and refines the requirements. Output from this phase includes a General Design document which describes to the functional office(s) what the system will look like when completed, and establishes a basis within which the administrative computing department (or the vendor) staff completes the Detail Systems design and subsequent development of the system.

The General Design phase is part of the Traditional Track for systems development.
The level of detail in a General Design document will vary depending upon the size and complexity of the system. In some cases, it may be appropriate to combine the General and Detail design phases.
The General Design is completed by the administrative computing department or a vendor, as appropriate. Extensive involvement of the functional office must continue through the General Design phase, to ensure that the system design meets the functional requirements identified during the Requirements Definition phase.

The outputs of this phase are: a General Design document, an updated Project Plan, and an updated list of the data elements (initially produced in the Requirements Definition phase as a preliminary list of elements).

b. General Design Document

The primary purpose of the General Design document is to describe the system from both technical and functional perspectives so that both the functional office and the application developers can understand what it will look like and do. The following sections may be included in the document.

Introduction. The introduction should reiterate the purpose of the system or analysis of the problem and the objectives to be met by the proposed system (or system modification), as stated in the Project Proposal and Requirements Definition documents.

Assumptions. Any technical or functional assumptions upon which the design is based should be stated. Also stated should be policy assumptions made during the General Design phase.
Differences from Requirements Definition. During the course of developing the General Design it is possible that some changes to the Requirements may be made. Such changes should be highlighted for review by the functional office.

Database Design. The design and structure of the database should be described. For example, if the database is relational, then the tables and the relationships between tables (including any required referential integrity) should be described.
How each data element fits into the database structure should be specified. For example, for a relational database, the data elements within each table may be listed, and the key fields identified.

Source(s) of Data. Sources of input data and method(s) for loading data (batch or online update) should be described. Required edits for data elements should be listed.
For interfaces from other MCDA sites or from external sources, consultation with the external organization must have occurred to ensure their willingness and ability to supply required data within the required timeframes. An outline of future involvement by the external organization, including providing test and production files, should be specified, either in the Design document, or in the Project Plan.

Processing Logic. Processing logic for batch and online processing must be described. Processing logic includes, for example, loading data into the database via online update or batch processing, handling of input files from other sources (interfaces TO the system), producing output files for other applications (interfaces FROM the system), producing reports, providing access to data for inquiry or ad-hoc requests, etc. An explanation of error or exception conditions and how they will be handled should be included.

Reports. Any reports required should be described, including: frequency (daily, monthly, on request, etc.), general purpose, selection criteria, and data elements to be included.
Output Files. Outputs of the system other than reports should be described. This may be files required for interfaces to other systems, or possibly files required for providing ad-hoc access to data by the functional office.
Interfaces. Interfaces with other applications may be described under Sources of Data (for interfaces TO the system) or Output Files (for interfaces FROM the system).
Whether interface files are described under earlier sections or in a separate interfaces section, the following information should be included for each interface file:

- Will the interface be TO the system (i.e. the system will receive the data from another application), or FROM the system (i.e. the system will produce the data for another application)?
- The format of the interface file should be described.
- Are the interfacing systems external to MCDA, at another MCDA site, or local to the MCDA?
- What data elements are required for interfaces from the system?
- How will the interfacing system be tested?
- What is the proposed timing of the interface in a normal cycle?

Ad-Hoc Access to Data. If the proposed system will be required to support ad-hoc access to data by the functional office(s), then how such access will be accomplished should be described.
System Architecture. A description of the system architecture should be included, i.e., standalone PC-based, client/server, web-based, etc. If remote access to data is required (e.g., by functional offices at other MCDA sites), describe how this will be provided. Issues such as local printing capabilities, downloads of data for specific purposes, and other system architecture requirements should be addressed.

System Controls. Design considerations to satisfy system control requirements must be described. For example:

- for batch processing, record counts should be provided as needed (e.g., record counts of records contained on an input interface file vs. records updated, added or deleted to a database, or rejected for errors)
- any processing controls that are necessary to ensure accuracy of arithmetic calculations (e.g., verification that the sum of monthly totals equals the annual total) should be included.

Security and Audit. Security, audit, and privacy considerations identified in the Requirements Definition must be addressed. Data access restrictions should be described, including identification of restrictions of selected users to particular functions (e.g. inquiry, or limited update access), or to particular data.
Audit considerations should be addressed in this section. How the audit needs identified in the Requirements Definition phase will be satisfied should be described.
Conversion Planning. The process for converting existing data (either hardcopy or machine-readable) to the new system should be described. Also, the means of initially loading data should be described.

User Training. If training will be required for users within the functional office, then the training approach and timeframe should be outlined.
Documentation. The documentation to be provided with the system should be listed and described (also see the section on Documentation).

a. Data Elements

An updated list of data elements including definitions should be completed during the General Design phase. The data elements should be categorized by the major data entities identified (i.e., for relational databases, the table(s) in which each data element belongs should be identified).
At a minimum, the data element definitions should include the following information:

- Data Element Name
- Type of data element (numeric or alphanumeric)
- Format of the data element (e.g., for numeric fields, the number of digits allowed, such as 3v2 (meaning a number of the format nnn.nn))
- General description or definition of the data element
- If the data element is a coded value, then the possible codes and the corresponding interpretation of each code
- Programming name (i.e., the name of the field in the database)

Additional information, such as data editing requirements and source of the data element, may also be included in the data element definitions. Data element definitions will be finalized during the Detail Design phase.

b. Review and Approval

The Project Leader is responsible for determining when the General Design Document may be distributed. When distributed, copies of the General Design Document should be given to:

- Functional office(s)

- Those within the administrative computing department who must review the document
- Internal Audit (for high risk systems) Others, for informational purposes

For high risk applications, copies of the design should be given to Internal Audit, unless otherwise indicated during the review by Internal Audit at the Requirements Definition phase. Internal Audit review focuses on audit considerations identified in the Requirements Definition.
The functional office(s) may be asked for a formal acceptance of the General Design document, indicating that they agree that the design accurately reflects their needs.

2 DETAIL DESIGN (TRADITIONAL TRACKONLY)

a. Purpose

The purpose of the Detail Design phase is to expand and refine the system design outlined in the General Design phase. Included in this phase is the development of program specifications from which programs can be designed and coded, development of test plans, and identification of networking and equipment needs.
The Detail Design phase is part of the Traditional Track for systems development.

The Detail Design is completed by administrative computing staff, or by a vendor for vendor-developed projects.
The output of the Detail Design phase include the following:

- Detail Design document, including program specifications.
- Updated Project Plan
- External interface specifications, where applicable
- Data Element definitions (refined from earlier versions of the Data Element dictionary)

b. Detail Design Document

The contents of the Detail Design document should include the following information:
Introduction. The objectives of the system should be briefly reiterated. An overview of the system with a description of the major components of the system should be included.
Assumptions. Any assumptions on which the Detail Design is based should be listed. If any of the assumptions made in the General Design have changed, the changes should be noted and the reasons for the changes given.

System Overview. The system overview should include the following:

- A high-level description of the various portions of the system and how they interrelate description of the major databases and files Every data source identified
- Every major output identified, including: reports (with report identifier), and interface files

Databases and Files. Every database within the system must be described. The structure and data elements must be included. For relational databases, every table and the data elements within each table must be listed, as well as the relationship between tables, and any required referential integrity. Where useful, a picture of the database(s) should be included. Key fields must be identified, and indexed fields, both unique and non-unique, must be identified.

Any sequential files must be described. For sequential files, the data elements must be listed, the record format and length given, key fields identified (and whether or not the key fields are unique), and the sort sequence specified.
Flowcharts or Data Flow Diagrams. Where appropriate, flowcharts or data flow diagrams showing the overall processing should be included. These flowcharts should include programs, files, utilities, etc.
Program Specifications. Program specifications will be used to code each program in the next phase: Coding and Unit Testing. Each program specification must include a description of the program function, input/output files, reads or updates to databases, reports, and a description of the processing logic. Specifications for online programs must also include screens, and event-driven processing (i.e. processing triggered by actions performed online by the user, such as pressing certain keys or selecting options).
See the end of this section for a sample detail design program specification for a batch COBOL program.

Data Retention. Required data retention periods for reports (hardcopy, fiche, image, or disk copy), sequential files, and, if applicable, data residing in databases, should be listed.
System Control and Data Verification. This section should describe how the integrity of the data is maintained within the system, from program to program, from cycle to cycle, and from system to system where there are interfaces with other systems.
Any verification of data that must occur outside of programs should be described. For example, if the functional office is responsible for reviewing certain reports before a specific process can be performed, this needs to be explained.

Backup and Recovery. The methods and frequency of backups of the databases and any sequential files must be described. The method and frequency for creating backups for offsite storage should also be described.
Methods for recovery of data in the event of system problems must be outlined.

Conversion. Any necessary one-time conversion of data must be described in detail. If conversion programs are needed, they must be specified. Any potential conversion problems or issues should be identified.
Hardware and Software Resource Requirements. For any shared resources, such as shared servers or printers, the impact of the proposed system must be evaluated to ensure that adequate resources are available to handle the new system without adversely impacting existing systems. Any significant upgrades or new hardware resources required for the new system must be identified. The following types of resources should be considered:

- memory allocation requirements on any shared server; disk space usage on shared server machines;
- Client workstation requirements (e.g., for PC/Windows clients, minimum RAM required, disk space required, level of operating system software required, and application software required);
- printer requirements, including estimates of pages printed, and determination as to whether or not existing printers will provide sufficient speed for the output expected; graphics printing capabilities etc.;
- networking requirements, including: workstation networking capabilities required (cabling and software), requirements for a LAN or access to a file server, etc.; and
- Level of client application software and server application software.

c. External Interface Specifications

If there is an interface between the new (or modified) system and either another MCDA site or an external organization (e.g., a health insurance carrier), then specifications for this interface must be written. If the interface is FROM the other site or organization, then the specifications must be sufficiently detailed to enable the other organization to develop the interface file processing. If the interface is TO the other site or organization, then the specifications must provide sufficient information for the other organization to effectively use the interface file in their application. At this point in the development process, any general issues regarding the interface should have been resolved.
Interface specifications must include the following:

- Overview of the data required for each file specified, including a description of the data.
- Frequency of data submission (i.e., monthly, annually, etc.), effective dates of the submissions, and due dates (e.g., 5th working day of the month).
- A record layout for each file transmitted must be included.
- A list of data elements, including the format for each data element, must be included.
- Physical file characteristics, including the data set name, record length and record format

OR

- The file name to be transmitted for files to be transferred via ftp, and the target system for the ftp transmission.
- The procedure for notifying the receiving site when the file has been submitted.

d. Data Elements

An updated list of data elements definitions must be completed.

e. Review and Approval

The Project Leader is responsible for determining when the Detail Design Document may be distributed. When distributed, copies of the Detail Design Document should be given to:

- functional office(s) using the system
- those within the administrative computing department who must review the document
- Internal Audit (for high risk systems)

The functional office(s) may be asked for a formal acceptance of the Detail Design document, indicating that they agree that the design accurately reflects their needs.

11 PROGRAMMING AND UNIT TESTING (TRADITIONAL TRACKONLY)

a. Purpose

The purpose of the programming and unit testing phase of a project is to complete the design and programming for each database and each program or portion of the system. The application developer is responsible for testing each portion of the system as it is completed.

In some cases it may be appropriate to involve the functional office in testing portions of the system as they are completed (e.g., testing a screen as the processing for that screen is completed), whereas for other projects, functional office involvement in testing may be primarily during the System Testing phase.

The Programming and Unit Testing phase is part of the Traditional Track for systems development.

b. Programming

The design of the databases and programs or portions of the system is based on the Design document (either the Detail Design, or the General Design for those projects where the Detail and General Design phases were combined). The programming standards and guidelines in effect for the MCDA for the language being used for systems development must be followed.

c. Unit Testing

The application developer is responsible for determining what testing is required for a program. A test plan, itemizing the test conditions to be exercised during testing, may be developed. The test plan can be an informal list for use by the application developer only, or may be a more formal list if functional offices will be involved in testing or test verification.
In general, testing should address the following:

- every function performed by the program should be tested;
- for event-driven programs (e.g., online programs), every event should be tested; any error conditions should be tested; and
- "boundary" conditions should be tested (for example, if an online program is designed to handle up to ten entries on a screen, then a test of entering tenor eleven entries and a test of entering zero or one entry should be performed.)

Test results must be reviewed carefully. The review of test results may be completed by the application developer, by the Project Leader, by the functional office, or by a combination of these staff.

c. Review and Approval

Review and approval of programming and testing is the responsibility of the Project Leader. Upon completion of the Programming and Unit Testing phase, the system must be ready for the System Testing phase.

SYSTEM TESTING

a. Purpose

The purpose of performing System Testing for a project is to ensure that all portions of the system work correctly (and as specified) and work together. Often portions of a system are initially developed and tested as independent pieces, and the system tests must ensure that the entire system works.

b. System Test Plan

The test plan must identify the steps to be performed and the test conditions to be exercised in order to test the system. The plan should identify who is responsible for developing the test data, performing the testing, and verifying the results. Normally, the functional office will be heavily involved in developing test data, verifying test results, and testing any online portions of the system. For some projects, the functional office may be heavily involved in preparation of the test plan as well.
The test plan should also identify the criteria for determining when testing is completed, and the person responsible for determining the completion of testing. A schedule should be established for the steps identified in the test plan.

c. System Testing

The types of testing that might be included in system testing, depending on the application, are as follows:

- test of batch and online processes in conjunction with one another(e.g., perform online testing of data entry, and subsequent batch processing using the data entered online);
- Different cycles of any processing (e.g., daily vs. monthly, fiscal year-end closing processes etc.);
- volume testing;
- performance testing, e.g., to confirm that online response time is adequate;
- backup and recovery procedures;
- test procedures and processing for receipt of files from other MCDA sites or external organizations, and creation of files for other MCDA sites or external organizations;
- test of network interfaces (e.g., lpr printing, ftp transmissions, and any other client/server processes);and
- Test of workflow queue processing.

In addition, the following types of testing should be performed as part of system testing, if appropriate for the given application:

- testing of any conversion processes or one-time data load processes;
- Parallel testing, for systems that will replace all or part of an existing system; and test of interface processing, both for incoming interface files, and the creation of outgoing interface files.

These three types of testing are discussed in more detail in the following sections.
Conversion Testing. For systems which will replace all or part of an existing system, it is often necessary to code one-time conversion programs, to convert the data from the format used in the existing system to that of the new system. This one-time process must be tested as part of the system test. It is important to identify how the conversion process will be verified, i.e. how it will be determined whether or not the data was correctly converted.
Note: Often parallel testing is a valid approach for verifying that the conversion programs are correct. The steps for using a parallel test to verify conversion processing are listed below:

- run the conversion programs, to convert the data from the existing format to the new format
- run both the existing system and the new system, in parallel

- convert the processed data from the new system back to the old system's format, and perform a comparison of data from the old system to data from the new system (this requires a one-time backwards conversion process).

Parallel Testing. For systems which will replace all or part of an existing system, parallel testing is normally performed. Parallel testing usually involves continuing with the existing system for production purposes, and running the new system "in parallel", using identical input data. A parallel test plan should be developed, outlining the steps necessary to complete the parallel test. The plan should include the following:

- how input data will be duplicated for the new system;
- how verification of data and reconciliation between the outputs from the two systems will be accomplished;
- how any inconsistencies in outputs will be handled (i.e., are some inconsistencies acceptable, and if not, how will the discrepancies be resolved);
- areas of the system that will be parallel tested, and those that will not be parallel tested; and
- Methods for ensuring that test data from the new system does not interfere inane way with the continuing production of the existing system.

Interface Testing. For most systems, every interface to and from the system must be tested as part of the system test. These tests must be coordinated with those responsible for the other applications (particularly applications which receive data from the new system), so that appropriate testing will be performed in the other applications well. The methods for verification of the interface tests must be identified, both within the new system, and in those applications receiving and testing interface files. The number of tests to be performed must be identified (e.g., if different cycles need to be tested, or special year-end versions of the interface need to be tested).

Acceptance Testing. As noted above, the functional office(s) must be heavily involved in many aspects of system testing. The functional office(s) may be involved in preparing the test plan and preparing test data, and will most certainly be involved in testing any online portions of the system and in verification of test data.
In some cases it may be useful to handle the functional office testing of online portions of a system as a separate, identifiable portion of the System Testing, often referred to as Acceptance Testing. By somewhat isolating this portion of the testing, it may be easier for the functional office to concentrate on the online portions of the system, and ensure that these portions of the system work as specified. Often it is helpful to isolate the acceptance testing technically as well (in a separate testing region, for example), to enable multiple types of system testing to occur simultaneously.

The Project Leader may elect to require a formal signoff from the functional office(s) upon completion of Acceptance Testing.

IMPLEMENTATION

a. Purpose

The purpose of the Implementation phase is to move the system into production.

b. Description

Prior to the start of the Implementation phase, an Implementation approach must be established. For some projects, implementation is straightforward. For others, especially in the case where an existing system is being replaced, the cutover process may be quite involved, and often must be performed under strict time- constraints.
The Implementation Plan should include a schedule of dates and who is responsible for each cutover step. Steps may include the following:

- Moving the code into production (e.g., into central production libraries); Distributing executable code on workstations within the functional office(s), as required;
- Moving data into production databases, including any conversion processes for converting existing data into the new system;
- Setting up user accounts or id's and associated authorizations for access to production programs and/or data, as needed; and
- Any required announcements that may need to accompany the start-up of the system.

If the system provides the capability for direct access by the functional office(s) to the data for ad-hoc reporting, then the functional office(s) is responsible for ensuring that privacy regulations and policies are adhered to for any individually- identifiable data elements (see Section 2. for further information regarding Privacy).

DOCUMENTATION STANDARDS

a. Overview of Written Documentation

The following documentation, if applicable, must be completed for each application:

- **Operations Manual**

The Operations Manual contains information required for the production control group to run batch portions of the system, if any. The Operations Manual must be reviewed by the manager of the production control group. (The production control group is the group within the administrative computing department that is responsible for running batch portions of production systems.)

- **Systems Manual**

The Systems Manual is for use by those responsible for on-going maintenance of the system.

- **User Documentation**

The User Documentation contains instructions for the functional office(s) on how to use the system, and may be combined, if appropriate, with the functional office's internal procedures manuals. The User documentation is completed by the functional office(s) or the administrative computing department, or some combination of staff from these departments, and may consist of online help or a hardcopy manual, or a combination of the two.

For a vendor package (or a vendor-developed application), the User Documentation may be completed by the vendor, or the vendor in combination with MCDA staff.

b. Training

If training of staff within the functional office(s) is necessary for the newly implemented system or system modification, the training may be performed by the functional office or by the administrative computing staff, or by some combination of these departments, depending on the nature of training required. Planning for any training required must occur well before implementation of the system, so that staff in the functional office can be trained and ready to use the system at implementation. For a vendor package (or a vendor-developed application), training may be performed by the vendor, or the vendor in combination with MCDA staff.
Training, such as introductory overviews of the system or system modification, may also be held as needed for maintenance staff and for the production control staff. The Project Leader is responsible for coordinating any necessary training within the administrative computing department.

c. Operations Manual

Purpose of the Operations Manual. The Operations Manual contains information required for the production control group to run batch portions of an application. If the application does not include any batch processing, or if the functional office is responsible for the batch processing, then there is no need for an Operations Manual for the application.
The primary purpose of the Operations Manual is to document how and when to run the batch processes, describe any tasks that must be performed before or after batch jobs are run, and provide instructions on what to do when a job does not complete successfully. In addition, the Manual should give enough of a background on the application to provide some context for the batch jobs. The Manual also includes information useful for reference, such as lists of datasets and reports, and a description of all interfaces.

Although the Manual is intended for use by the production control group, this Manual may also be useful to those responsible for maintaining the system, and should be readily available to maintenance staff as well as the production control staff.

Contents of the Operations Manual. The Operations Manual is divided into two major sections:
Part 1 - General Information. Part 1 of the Operations Manual includes overall information about the application.
Part 2 - Running the Jobs. Part 2 of the Operations Manual documents the individual job procedures within each Operations Stream.
The contents for both parts are described below.

Part 1 - General Information

The first part of the Operations Manual provides overall information about the application. The following sections should be included:

1. System Description

The System Description section should contain a brief overview of the application. This overview can be taken from other documents, such as the Requirements Definition or System Definition document, or the Systems or User Manual.

2. Application Control Information - Project Contacts

A list of the contacts for the project should be listed, including contacts for: production control

3. Processing Schedule

Depending on the application, scheduling of jobs may be straightforward (e.g., "run monthly as soon as input data is available"), or quite complicated. For example, scheduling jobs for the Payroll system is complex, both because of the number of different jobs and their interrelationships, and because of the interaction between the jobs and the processing in the functional office.
This section of the Operations Manual should give an overview of how scheduling of jobs is accomplished. For applications where scheduling is complex, details for how to determine the scheduling for jobs must be described, and who is responsible for producing the schedule must be stated (e.g., the functional office, or production control in conjunction with the functional office).

4. Interfaces TO the Application

All interfaces must be documented. The following information should be included for each interface file sent TO the system:

- a brief description of the interface;
- what location or locations the interface file is from (e.g., MCDAs, another application run locally, or an outside vendor) and, for interface files from applications run locally, what application the file is from;
- which job or jobs use the interface file;
- The schedule for receipt of files (for files sent on a regular basis, this schedule is usually expressed in terms of number of workdays from the beginning or end of a period of time. For example, for a monthly file, the file may be due the 5th working day after the end of each month.);
- The method for transmitting the file (e.g., the file will be transmitted electronically via ftp to a specified target machine, or the file will be transmitted on tape medium, etc.);
- The name of the dataset, for whatever target medium on which the file will be sent (e.g., for files transmitted to a local MVS machine, the MVS dataset name (DSN) is required);
- Acceptable values for any portion of the dataset name which is variable, e.g., acceptable format for cycle parameters such as MMMYY (e.g.,FEB98);
- Any file characteristics that must be known in order to process the file (e.g., for files transmitted to a local MVS machine, the record length (LRECL) and record format (RECFM, i.e. FB (fixed blocked) or VB (variable blocked)) are required);
- The retention period to be assigned to the file; and
- A list of other information to be provided by the sending location when the file is sent, such as record counts for the file.

5. Interfaces FROM the application

All interfaces must be documented. The following information should be included for each interface file created by the application:

- a brief description of the interface;

- what location or locations the interface file is for and, for interface files created for applications run locally, what application is receiving the file;
- which job or jobs produce the interface file;
- what the target schedule is for production of the interface file(e.g., by the 10th working day of the month);
- the method for transmitting the file to the target location (e.g., the file will be created on a local machine, for locations to "get" electronically using ftp, or the file will be transmitted on tape medium, etc.);
- The name of the dataset, for whatever target medium on which the file will be sent (e.g., for files created on a local MVS machine for locations to "get" using ftp (file transfer protocol), the MVS dataset name (DSN) is required);
- Acceptable values for any portion of the dataset name which is variable, e.g., acceptable format for cycle parameters such as MMMYY (e.g.,FEB98);
- Any file characteristics that must be kept constant for other locations to be able to process the file (e.g., for files created on a local MVS machine, the record length (LRECL) and record format (RECFM, i.e. FB (fixed blocked) or VB (variable blocked)) must be kept the same for other locations to "get" the files for processing using ftp);
- The retention period to be assigned to the file; and
- Information regarding how notification to the recipient of the file is handled, including any information to be provided, such as record counts for the file.

6. Account Information

The account(s) used by the production control group for this application should be described.

7. Datasets

A list of datasets used by the application should be included. Depending on the application, the list may be split into the following categories:

- libraries (e.g., on MVS, the libraries used for the application-specific software);
- disk datasets (include dataset name and description; this list may be split by machine, for applications where multiple machines are used);
- onsite tape datasets (include dataset name and description); and
- Offsite tape datasets (include dataset name and description).

8. Reports

A list of reports should be included, with the Report ID, the Job that creates the report, and a brief description. The list should be in order by Report ID.

9. Security

Security within applications can be quite complex, and a complete explanation of all security used within an application may not be covered completely in the scope of this Manual. However, the overall rules for the application related to security that might affect the production control group, will be stated in this section.

10. Console Logs

The retention period for the hardcopy or online console logs should be stated. Instructions for accessing console logs should be included. (Note: unless otherwise stated, the retention period for all hardcopy and online console logs is 13 months.)

11. How to Update this Manual

Information must be provided on where the machine-readable version of the manual resides, and how to update it and print copies. People who should be given a copy of the manual after revisions are made should be listed. Generally, the list should include: the production control contact and the maintenance contact. In some cases, the list may include one or more functional office contacts as well.

12. Glossary of Terms Used(optional)

A glossary of commonly-used terms that are application-specific may be included. For example, the term "monthly maintenance" within the Title Code System (TCS) refers to the online updates performed by the users on a daily (or as needed) basis -- having a definition of this term in a glossary could be useful for the production control group.

13. Transmission

Any files or reports that are transmitted to other MCDA sites (or external vendors) or made available for pick-up electronically by other MCDA sites (that are not already described under Interfaces) must be listed, including:

- the job that transmits the file or report or makes it available for pickup; and
- The dataset name of the file or report.

Part 2 - Running the Jobs

The second portion of the Operations Manual includes instructions for running each batch job within the application. The batch procedures to be run for the application must be organized by Operations Streams. Each Operation Stream is a logical set of batch procedures that are normally run together. A smaller system may have only a single Operation Stream, while larger applications with extensive batch processing may have a number of Operation Streams. An Operation Stream may consist of only a single job, or many inter-related jobs.

Examples of Operation Streams include: Payroll Monthly compute, and General Ledger monthly processing.
A list of all Operation Streams in the application should be included at the beginning of Part 2 of the Manual. Also, a brief description of the Operation Stream should be included at the beginning of each Operation Stream.
Within each Operation Stream, there should be a separate section on each batch procedure or job. The following sections should be included, as needed, for each process:

1. Description

A brief description of what functions the job performs (e.g., updates the primary database, produces reports, creates interface files, etc.)

2. Frequency

The frequency of the job (e.g., monthly, upon request, etc.).

3. Programs Executed

A list of the programs executed in the job, with a brief description. For MVS jobs, the JCL Procedure and Job step should be included.

4. Files

The files used in the job, including Input Files (i.e. read-only files), Created Files, and Updated Files.

Note: This list of files should include databases that are read or updated by the job, even in cases where the use of the database may not be obvious from looking at the job itself. For example, DB2 tables that are read or updated from an MVS batch job do not appear in the JCL, but should be included in the list of files.

5. Resource Requirements

Resource requirements might include, for example, execution time (particularly if the job is long-running), tape drives, and lines of printed output.

6. Pre-Processing Instructions

- **Dependencies**

Any dependencies for this job must be listed. Examples are: confirmation that another job has successfully completed (where confirmation may be performed by the production control group or by functional office contacts), receipt of data from other systems, or status of a CICS region (e.g., a specific CICS region must be down before the job can run).

- **Prompted Variables**

All variables that are prompted or required as part of submission of the job must be listed and described. For example, if a cycle date is required, the Manual should show the input format (e.g., MMMYY where mmm is JAN, FEB, etc.), and describe how the value of each variable is derived (e.g., use the current month, use the prior month, the value must be supplied by the functional office contact, etc.).

7. Post-Processing Instructions

Instructions for tasks to be performed after completion of the job. These tasks may include, for example:

- Verification procedures (checking of return codes, record counts, totals, etc.)
- Log entries (record counts, volume serial number of tape to be sent to an outside vendor, etc.)

- fiche report handling
- interface file handling (for files created by the application and sent to another system)

8. Report Distribution

Instructions for distributing reports. All reports to be distributed must be listed (report id, and dataset name (if the report is retained on disk)), with the number of copies and the person to whom the report is sent.

9. Rerun Instructions

Rerun and restart instructions. The instructions should include information on what to do when the job ends abnormally in any given step, and what to do if a complete rerun is required even when the job has successfully completed. Instructions may include, for example, information on restoring databases or files (this may refer to a separate job); deletion of created files before rerunning; or contact the maintenance programmer under certain circumstances.

d. Systems Manual

Purpose. The intended audience for the Systems Manual is the programmer(s) responsible for maintenance of the application. The Systems Manual must provide a technical overview of the system, and must provide details on technical approaches taken in the application that are not readily apparent from the code.
Code as Documentation. One of the primary sources of information for any maintenance programme is the source code itself. Application developers are responsible for writing clear code, and for including comments in the code. To the extent possible, every program should have comments at the beginning describing the functions the program performs, and giving other overall information. Throughout the code, comments should be included to assist in following the flow of logic, and to describe any particularly complicated logic.

The Systems Manual is not intended to repeat information that can be found easily in individual programs. Instead, the Systems Manual should provide overall information and technical details regarding the application that would be difficult or impossible to discern by looking at individual programs.
Contents of the Systems Manual. Since applications vary widely in technical approaches, it is the Application Developers' and Project Leader's responsibility to ensure that the Systems Manual contains information that will be useful to the maintenance programmers and that the Systems Manual is complete. The following sections are proposed as guidelines.

e. Application Description

This section provides a brief overview of the application. (This overview may be taken from the Requirements Definition or System Definition document.)

f. Technical Environment

This section provides a description of the technical environment, including:

- the platform or platforms upon which the system runs (i.e. Operating System, and, if the application is dependent on specific hardware, the hardware required);

- The application software used for developing the system on each platform, including programming languages and/or application development tools; and
- any utility products required for each platform, if these utility products are specific to the application (utilities that can reasonably be assumed to be available given the platform need not be listed; for example, there is no need to list a "sort utility" on MVS, since one can assume such a utility is available).

g. Databases

The databases must be identified, and the database structure described. An overview of each database and/or file should be provided, and the relationship between databases or files should be described. For relational databases, the Create Table SQL code may be used to provide the details of each table. In addition, the referential integrity must be described, and the primary keys identified (either in the SQL code, or outside of it, depending on the code).

h. System Flow charts

If available, system flowcharts showing the flow of data and/or processes should be included. (These flowcharts may be included as an Appendix.)

i. Glossary of Terms

Any application-specific terms that are commonly used should be defined. For example, the term "primary appointment" in the Corporate Personnel System (CPS) might be defined in the glossary.

j. Application Interfaces

All application interfaces, both FROM this application to other applications, and TO this application, must be identified and described. For interface files produced by the application, the job(s) producing the interface file should be identified. For interface files received from other applications, the dependencies on each interface file should be described.

k. Security and Authorization

The methods used for providing security and authorization for the application should be described. This may include, for example, database security (e.g., DB2 or Sybase database permissions), dataset security (e.g., RACF protection of non-DB2 files in MVS), or process security (e.g., RACF protection of CICS resources, or internal application-specific security to allow only select users to access certain processes, such as update access to the database).

l. General Technical Approach

Information regarding the general approach taken for coding the application should be described. For example, an MVS/CICS system may have a uniform structure for each of the online programs, and a uniform approach for the relationship between programs and CICS transaction ID's. In this case, this uniform generic structure of the programs should be described.
Naming conventions should also be described. Often there are application-specific naming conventions that, once explained, can help during maintenance.

Any other overall technical approaches or uses of technology that would be helpful to know when maintaining the system should be included. (Note: this section is the most difficult to define, since it varies dramatically for each application. This section is also one of the most potentially useful and important sections. It is the responsibility of the Application Developer(s) and the Project Leader to ensure that this section provides reasonable and useful information.)

m. Ad-Hoc Access to Data

If end-users will be provided ad-hoc access to data within the application, then the method for providing this access should be described. Also, any restrictions imposed on the application due to the fact that end-users have direct access to the data should be described. For example, if modifications to specific portions of the database could affect the end-users' view of the data, then this should be noted, so that any database changes made due to system maintenance will be coordinated with end-users.

n. User Documentation

Purpose. The purpose of user documentation is to provide the functional office(s) that use the application with information on how to effectively use the application, or use the data provided by the application. The content, style, and extensiveness of user documentation will vary greatly from application to application, depending on the size, complexity, and user interaction for each application. The sections described below are guidelines for the types of information that may be contained within the user documentation. It is the responsibility of those who are writing the documentation to ensure that the information provided is useful and relevant.

Some applications include varying levels of access for different groups of users. If the difference in access is extreme, then it is probably appropriate to create separate user documentation for the different groups of users. For example, if an application allows online update for a small number of users in a single department, and allows online inquiry for a large number of users across many departments, then it would be best to have separate documentation for the update portion of the application and for the inquiry portion of the documentation. With separate documentation, each user group could be provided with only the documentation applicable to their needs.

For operational systems it may be appropriate, in some cases, to combine the user documentation for an application with the functional office's internal procedure manual. In this case, the functional office will need to either take the lead or be heavily involved in writing the user documentation, so that it will be correctly integrated with the documentation for their internal procedures.
User documentation may be provided using different formats, or a combination of formats. For example, for online update or inquiry, online help may provide sufficient information for user documentation. For other applications, a hardcopy manual may be most appropriate. For many applications, a combination of the two (online help plus a hardcopy manual) works best.

Contents of the User Documentation. The following sections include topics that would normally be found in user documentation for an application.

o. Overview of the Application

An overview of the application should be provided. This can often be taken from the Requirements Definition or System Definition document. The overview should include a description of the purpose of the application, and some information regarding the functional office or offices using the application.
This overview should also review the major processes of the system, and describe how they interrelate.

p. General Technical Environment

While functional office users should not need to know any details about the technical environment in which an application runs, it is often helpful to know a little about the technical environment. For example, for an application with a client/server architecture
where the client side operates under PC/Windows and the server side provides the database, it is useful for users to know that the client portion runs within the familiar Windows environment, but also to understand that updates to the database occur in a central, shared database, and any changes made to the data will be available immediately to other users.

q. Online Update

Users with online update capabilities in a system must have sufficient information to correctly update the database(s). Often the information required for using a particular update screen may be incorporated into text on the screen itself, or in online help available from the screen. The information available online may include, for example, valid code values for individual fields; information on which fields are required; and other edit information, such as edits involving more than one field.
In addition to information required for individual screens, some overall information on updating the database may be required, depending on the complexity of the update process. For some applications, the user may need to understand some of the interrelationships of the data in order to update the database. For example, it may be that a department must exist as an entry in the database before an employee in that department is entered into the database.
Any features that are available across screens may be described in the user documentation. For example, if there is an application-specific "copy" feature, that allows the user to copy data from an existing source before modifying the data and updating the database, then this could be described in the user documentation.

r. Batch Processing

While functional office users may not need to understand the details of any batch processing included in the system, any interdependencies between online update, data retrieval, and batch processing must be described. For example, if a financial system requires that data entered online must balance to zero before a batch process can proceed, then this needs to be described. Another example is when users are responsible for reviewing data provided on a standard report to confirm its accuracy before batch processing can proceed. Any such control points, where there are dependencies between actions taken by the functional office (whether it is online update, review of data, or manual procedures) and batch processing must be clearly documented.

In addition, any post-batch processing that is the responsibility of the functional office must be documented. For example, if the database is updated through a batch process, and an edit report with warning-level errors is produced for review by the functional office and possible subsequent data corrections, then this must be documented.
In some cases, the functional office is responsible or shares responsibility for scheduling processing for an application. This is particularly critical for operational systems where internal procedures within the functional office affect the timing of processing within the application. If scheduling is either partly or totally the responsibility of the functional office, than this should be included in the user documentation.

s. Data Integrity

Any system controls built into the application to ensure data integrity is preserved should be described in the user documentation. Depending on the nature of the system controls, this topic may be incorporated into the sections on online update or batch processing. An example of one type of system control mechanism is a regular edit report, run to check certain aspects of the database. The user documentation should explain the purpose of such an edit report, and describe how to recognize if there may be potential data problems based on the information in the report.

t. Retrieval of Data

Retrieval of data from the system for use by the functional office(s) may be through a variety of methods, including: routine production of standard reports, online inquiry to specific data, or ad-hoc inquiry or reporting.
For standard, production reports and standard online inquiry screens, the users must be provided with information on data included in the report or on the screen. This can often be provided through the Data Element Dictionary. If some of the data displayed on a report or an inquiry screen is derived (rather than simply displaying data available in the database), and the derivation is not obvious, then these derived fields must be documented.
If there are any timing issues which may affect the data provided to users on standard reports or inquiry screens, than these issues must be explained. For example, if data in the database comes from multiple sources, which may sometimes result in data within the database having somewhat different effective dates, then this should be explained.

If functional office users have the capability to perform ad-hoc inquiry and reporting using some or all of the data within the application, then the users must have an understanding of the structure of the database and interrelationships of data as well as definitions for individual data fields. The user documentation must provide sufficient information regarding the data available for ad-hoc inquiry and reporting for the functional office users to make effective and accurate use of the data. If users have access to only one simple file (or table), then this may be straightforward to document. If users have access to multiple, complex databases, then this documentation must be far more extensive.

u. Data Element Dictionary

The data element definitions established during system development must be made available to the functional office users. This dictionary may be incorporated into a user manual, or may be made available separately.

v. Authorization Levels

If the application includes different levels of authorization for users, then each level and the capabilities available for each level must be documented. For example, if most users of an application have update access to the main database, but another level of authorization allows update access to specific tables used within the application, than these different update capabilities must be described.

w. Problem Resolution

Information on who to contact in the case of a problem arising should be available as part of the user documentation. This contact person could be someone within the functional office responsible for overall coordination for the application, or may be someone within the administrative computing department.

POST-IMPLEMENTATIONREVIEW

a. Purpose

The purpose of the Post-Implementation review phase is to:

- Evaluate the system in terms of its success at satisfying the functional needs described in the System Definition or Requirements Definition document;
- Ensure that production processes are going smoothly;
- Ensure that procedures are in place to handle on-going maintenance of the system; and
- Evaluate the system for inclusion in any local Contingency/backup Plan.

This phase also provides an opportunity for reviewing the systems development process, and identifying possible improvements for future system development projects.

b. Description

The method for performing the post-implementation review is at the discretion of the Project Leader and the Director of the administrative computing department. The review may be informal, or may be conducted more formally, depending in part on the scope of the system.

ANNEX 7: INTEROPERABILITY

By interoperability we mean here the capability of (two or more) systems to exchange seamlessly data, information, and knowledge.
E-Government interoperability has become a crucial issue because recent ICT investments have reinforced the old barriers that made government decision-making, not to mention citizen access to public services, difficult. In a number of governments, agencies are deploying new ICT systems with specifications and solutions relevant to their particular needs but without adequate attention to the need to connect, exchange and re-use data with other agencies' ICT systems. The result is a patchwork of ICT solutions that is not always compatible with each other and an e-government programme that does not meet its goals of better decision making; Better public services; Better governance.

Recognizing that e-governments should be transformative and become more citizen – rather than government – focused in delivering public services, investing in the development of an e-government interoperability framework is fundamental. Otherwise, the millions of dollars spent on e-governments would rarely lead to good governance.

E-Government interoperability enables one-stop, comprehensive online services for citizens and businesses by linking the diverse services that are offered by different agencies. Furthermore, increasing the ease at which information is shared among individual agencies (up to the point allowed by law) makes for better and/or new services. For instance, the administration of justice would be faster and more effective if the information systems of various agencies under the criminal justice system (police, public prosecutors, public attorneys, courts, prisons) were able to share data.

E-Government interoperability can be achieved through the adoption of standards. Achieving interoperability through standards usually entails adopting a Government Interoperability Framework (GIF): a set of standards and guidelines that a government uses to specify the preferred way that the agencies, citizens and partners interact with each other.

The GIF includes "the basic technical specifications that all agencies relevant to the e-government implementation should adopt." The e-GIF covers communication with the general public, not just communication within government or between applications. It aids the exchange of information between government systems and the interactions between:

i. Government and citizens
ii. Government and intermediaries
iii. Government and businesses (worldwide)
iv. Government organizations
v. Government and other governments

The GIF has been derived from the Government Enterprise Architecture (GEA): a strategic planning framework that relates and aligns government ICT with the governmental functions that it supports.

The Government agencies shall endeavor to adopt the open standards to enable interoperability. Open standards play a key role in achieving interoperability. Open standards enable products to work together. They also lead to diversity of suppliers/vendors and technological development. Open standards also ensure quality.

The main characteristics of open standards:
Availability: Open standards are available for all to read and implement. Maximize end-user choice: Open standards create a fair, competitive market for implementations of the standard. They do not lock the customer into a particular vendor or group.

No royalty: Open standards are free for all to implement, with no royalty or fee. Certification of compliance by the standards organization may involve a fee.

No discrimination: Open standards and the organizations that administer them do not favour one implementer over another for any reason other than the technical standards compliance of a vendor's implementation. Certification organizations must provide a path for low- and zero-cost implementations to be validated, but may also provide enhanced certification services.

Extension or subset: Implementations of open standards may be extended or offered in subset form. However, certification organizations may decline to certify subset implementations, and may place requirements upon extensions.

Predatory practices: Open standards should employ license terms that protect against subversion of the standard by embrace-and-extend tactics. The licenses attached to the standard requires the publication of reference information for extensions, and a license for all others to create, distribute and sell software that is compatible with the extensions. An open standard should not prohibit extensions.

The Government Agencies shall ensure that Open standards should have the following minimum characteristics for a standard to be open:

- Easy accessibility for all to read and use;
- Developed by a process that is open and relatively easy for anyone to participate in; and
- No control or tie-in by any specific group or vendor.7 Open standards help ensure interoperability

Content of the GIF

The GIF will allow government to manage information better and provide services to citizens, businesses and other stakeholders in an integrated manner. It includes the technical specifications that all agencies involved in the e-government implementation should adopt.

The GIF that will support the Shared Government Platform (SGP) will include:

a) Context;
b) Technical content;
c) Development process;
d) Implementation; and
e) Compliance regimes.

Context will include: Definitions, aims, objectives, principles, background, audience, benefits and relationship with other initiatives.
Technical content will include: Key technical policy statements, standards, standards categories, standards selection criteria and standards status.
Development process will include: Development and revision process, actors and responsibilities, and mechanisms for consultation.
Implementation will include: Implementation support tools
Compliance regimes will include: Interoperability indicators, responsibility for compliance, stakeholders, guide tools and non-compliance.

INTEROPERABILITY FRAMEWORK

1. Shared Government Platform (SGP)

It is a standard that guides the repository of data and information that can be accessed, shared and used by government agencies. In the process, the government agencies enhance their efficiency – and save money – by sharing resources, eliminating duplication of work, collaborating, and exchanging knowledge and experiences.
The SGP aims to provide a whole-of-government approach to the diverse operations of agencies, while at the same time recognizing both their unique and common features.

Government agencies do interact with each other, but this interaction relies heavily on the compatibilityoftheirsystemsandapplicationsandtheinclinationoftheirmanagementsto share or exchange data and information.
Currently, there is no common platform for the entire government where agencies can share resources or data. Some agencies maintain their own databases and there is no systemthatconsiderstthewholeofgovernmentinsharingcredibledataandinformationon demand.
How the SGP will work
The business problem the system intends to solve is: "How will government agencies share credible data and relevant information efficiently with each other on demand?"

Providingasinglesystemofsharinganddisseminatingdataandinformationforthewhole of government is problematic and next to impossible because of the varied interests and functions of government agencies. However, this can be overcome by organizing government agencies into collaborative groupsbasedonthetypesofdataandinformationthatarerelevantandmeaningfultoeach ofthem. Turningparticipatingagenciesintofederatedcollaborativegroupswillmakethe system easier to manage and implement. Each collaborative group can focus on valuable sharing or exchanges. Agencies in the collaborative groups will be required to customize the SGP standards to their environment while ensuring that all the components are catered for.
The agencies through this standard they shall endeavour to bring on board all agencies that they collaborate with in delivering services to the citizen to avoid duplication.
Technical Architecture

Section3detailsthetechnicalarchitectureandcorrespondingtechnologyenvironmentand specifications.Thescopeofthestandardistoenabledatasharingandexchangefor the "whole of government."

The general technology environment specifications detailed in this section will apply to the development, staging, and production environments.

2. Section 1 – General Systems Description

The current state of the government's ICT environment is a patchwork of expensive systems built without the foresight of interconnectivity.

In the effort of respective MIS groups to "deliver" on their commitment to software-enable their processes, most systems were intentionally implemented as a stop-gap to meet the specific objectives of the implementing agency without considering how it would affect the requirements of the whole of government.

If we were to liken this to a room in a building, the wall separating it from the rest of the building would be riddled with holes and exposed wiring and plumbing from connections made to the room that were not considered in its initial design.

The amount of after-the-fact independent connections to the different systems of an agency negatively affects the performance of the system and, more importantly, the security of the system – the higher the number of independent nodes, the higher the number of points where failure and security breaches could occur.
The complexity needed to attain interoperability between government agencies becomes apparent when we factor the number of unique elements characteristic to each government agency and government-owned corporations. A relatively simple process such as registration can be highly complicated when we consider all of these variables into a proposed solution.

To reduce the work needed to harmonize differences in areas as simple as registration, most fall into a thinking process that gravitates toward oversimplification. Oversimplification happens when system architects or system designers attempt to enforce a single view of process and meaning on a group of entities that perceive and comprehend the world differently. For us to systemically design a sustainable solution, it is important to embrace the natural complexity that characterizes governmental operations, and use this reality to define a systems model that will enable us to architect an effective solution specific to government.

3. The Universe of Government

Viewing the government as a cosmos of enterprises makes it easier for us to appreciate the impracticality of forcing a single, common model that controls the composition, function, semantics, or mode of interaction between interacting agencies and agencies with the general public. Attempting to do so will be an effort in futility for the simple reason that such a model does not exist in the physical world. What does exist in nature, however, are systems of unique and independent wholes operating seamlessly within the larger whole or systemic universe.
That said, the most practical approach is to factor the co-existence of a multiplicity of methods, functions, processes and standards as a specification of a unified cosmos of meta- enterprise systems.
The unified cosmos of meta-enterprise systems' role and function is to set the context and framework on how enterprises within this system would interact and thrive amidst diversity and variety without intruding into the different enterprises that comprise the macro system.
The systems environment of each agency within this unified cosmos of meta-enterprises are made up of a combination of "conventional" elements such as: Transactional Systems and Data (including financial and regulatory), Administrative Systems and Data, Supply Chain, Logistics and Inventory Systems and Data, Product and Service Catalogue Systems and Data, Shared Services Systems and Data, Non-Structured Systems and Data, Nomenclature Dictionary Systems and Data, Performance Management Systems and Data, Industry and Function Specific Referential Systems and Data, Decision Support and Business Intelligence Systems and Data, Security Management Systems and Data, Workflow Management and Routing Systems and Data, and Central Repository Systems and Data.– each component functioning as wholes that form part of the greater whole (universe).

The state of deployed systems within each enterprise, driven by the functions and processes unique to each business unit within that government entity, differentiates one entity from another.

Each differentiating feature drives the level of complexity of processes, which in turn determines the complexity of effort involved to integrate systems within the intra-enterprise and the extra-enterprise. The more unique the process or function, the higher the degree of complexity, and the more difficult it is to integrate or interoperate with external entities.

4. Conventional Functions and Processes

Human resources, financial management, accounting, budget, and procurement, to name a few, are common functions across all government entities. It is in these areas where standardization and interoperability will yield higher levels of efficacy since the processes of each of the underlying functions will vary little between government entities. For example, standard accounting processes are governed by the National Government Accounting Standard and procurement processes by the Procurement Law.

5. The Context of Standards

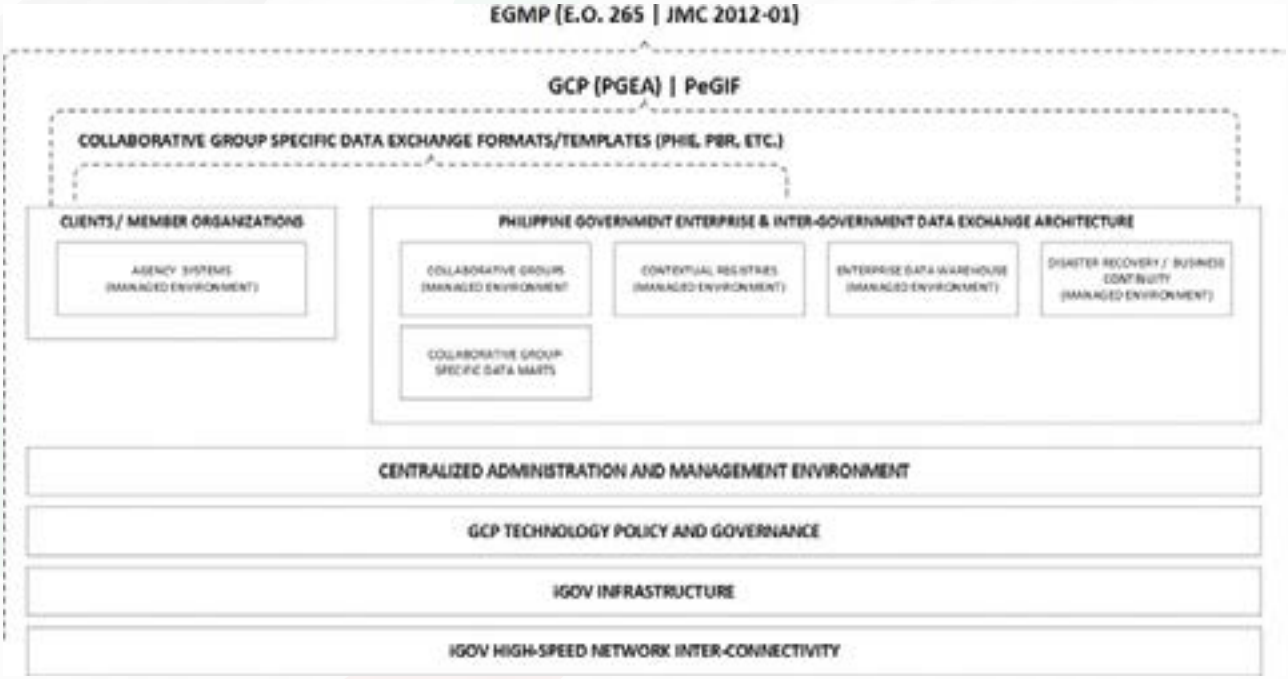


Figure - The Context of Standard of the Government Common Platform

6. System Variety

The closer the affinity of a process to frontline, regulatory, and enforcement services, the more difficult it is to create a standard across the meta-enterprise.

The level of variety increases as the relative proximity of a function or process to frontline public service processes decreases. This is a material fact given that frontline, regulatory, and enforcement services are required to manage transaction level details of interactions with clients with varying levels of need, context, comprehension, culture, worldview, and behaviour. This is why the process required to secure medical-related permits is different from the processes required to secure a driver's license, a passport, a social security number, a tax identification number, a business registration, or an importation permit.

7. The Government Universe of Meta-Enterprises

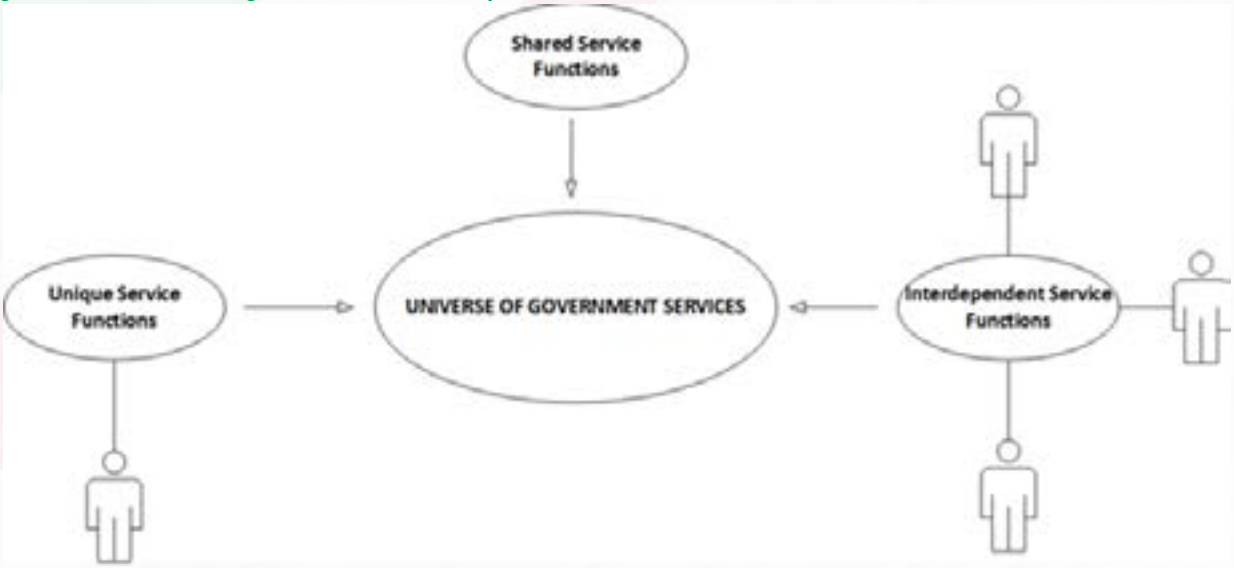
The services of Government entities have been categorized into three primary categories: 1. shared, 2.unique, and 3. interdependent. Please take note that what we have categorized are not "processes" but "services."

a. Shared services are the common functions (regardless of organizational context) involving human resources, financial management, etc.
b. Unique services are those involving frontline regulatory and enforcement. And

c. Interdependent services are those that require input from one or more external entities to complete the delivery of the service (e.g., Philippine Business Registry and Philippine Health Information Exchange). Each service category will, by necessity, maintain a set of attributes, data, and information requirements specific or unique to that service category.

These categories are rudimentary to defining the Government's interoperability and enterprise service-oriented architecture (Soak) model. The future whole of government enterprise shall provision a SoA enterprise as a standard for applications that enable shared service functions, and applications that enable interdependent service functions.

Figure 2 - Context Diagram of Relationships in the Universe of Government Services



Shared Service Functions

The set of shared service functions (whole of government perspective) that will be enabled by a centrally-managed enterprise system is essential to reduce unnecessary complexity in designing the meta-enterprise and in managing the future technology environment.

Shared service enterprise applications will initially include:

- Identity Management System
- Enterprise Information Portal
- Unified Communications (Email/Instant Messaging/Voice)
- Enterprise Calendar
- Document Management and Archival System
- Human Resource Management and Payroll System
- Accounting and Budget Management System
- Supply and Inventory Management System
- Strategic Sourcing and Procurement System
- Fixed Asset Management System
- Facilities Maintenance Management System
- Logistics and Transport Management System
- Electronic Payment System
- Contracts Management System
- Enterprise Planning
- Project Management System
- Content Management System
- Performance Management System

Unique Service Functions

Unique services are those that require the agencies to capture, track, and resolve the state and relevance of data on case-by-case basis as prescribed by their charters. The lifecycle and the disposition to "share" data or information, given their inherent sensitivity, are determined by the "trustee "agency.
These services include the likes of law enforcement, national defence, national intelligence and security, department of justice, court systems, and so forth. Noticeably, the key characteristic of data generated by these functions is the level of sensitivity. They typically fall under the category of "personally identifiable information" – which provides deep insight into the private life of a subject or constituent? As such, these types of data, as a rule, are not universally shared even between government agencies.

The parameters for "sharing" and "exchanging" data between agencies entrusted with sensitive data are defined within a legal framework between interacting agencies. It is within this legal framework where the scope of data to be shared, along with the method and timing of sharing or exchange, are explicitly defined.

Interdependent Service Functions

Interdependent services refer to functions performed by frontline government agencies that produce a set of data, such as in the case of business name and entity registration, certificates of registration, and issuances of permits and licenses at the national and local levels. Simply put, it involves multiple government agencies sequentially approving the state of data until the data arrives at a completed or final state.

Interdependent functions are a closed-loop process. In this environment, data work its way through a predefined lifecycle – from cradle to grave – until it arrives at a state deemed final or complete.

The tight coupling between data states as determined by interacting agencies means that the effectiveness and efficiencies of the whole are dependent on the efficiency of every component within this closed loop system. Any form of latency or delay that occurs at any given point in the process negatively affects the whole. Faster processes will create congestions against slower processes.

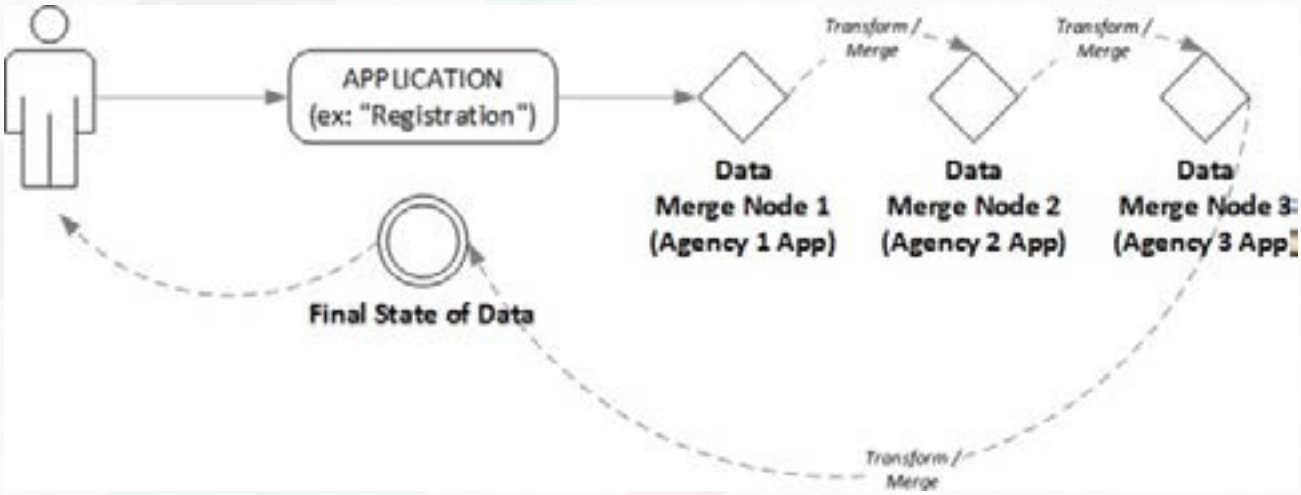


Figure 3 - Data Transformation in an Interdependent Service Environment

The agencies concerned need to consider, agree and document the Performance latency factors in the exchange of data that are bound to affect the state of data as it hops from agency to agency. Among these are the bit architecture, technology architecture, technology platform, run-time environment, data schema, data types, application-specific proprietary protocols, and security protocols, to name just a few. Latency is certain to occur even if only one of these condition sexists.

A practical and proven way to regulate disparities in data or in processing capabilities between the systems of interacting agencies is to introduce a system "regulator" in the form of an enterprise service bus. This enterprise service bus or ESB is at the heart of a service-oriented architecture environment (see Figure 4).

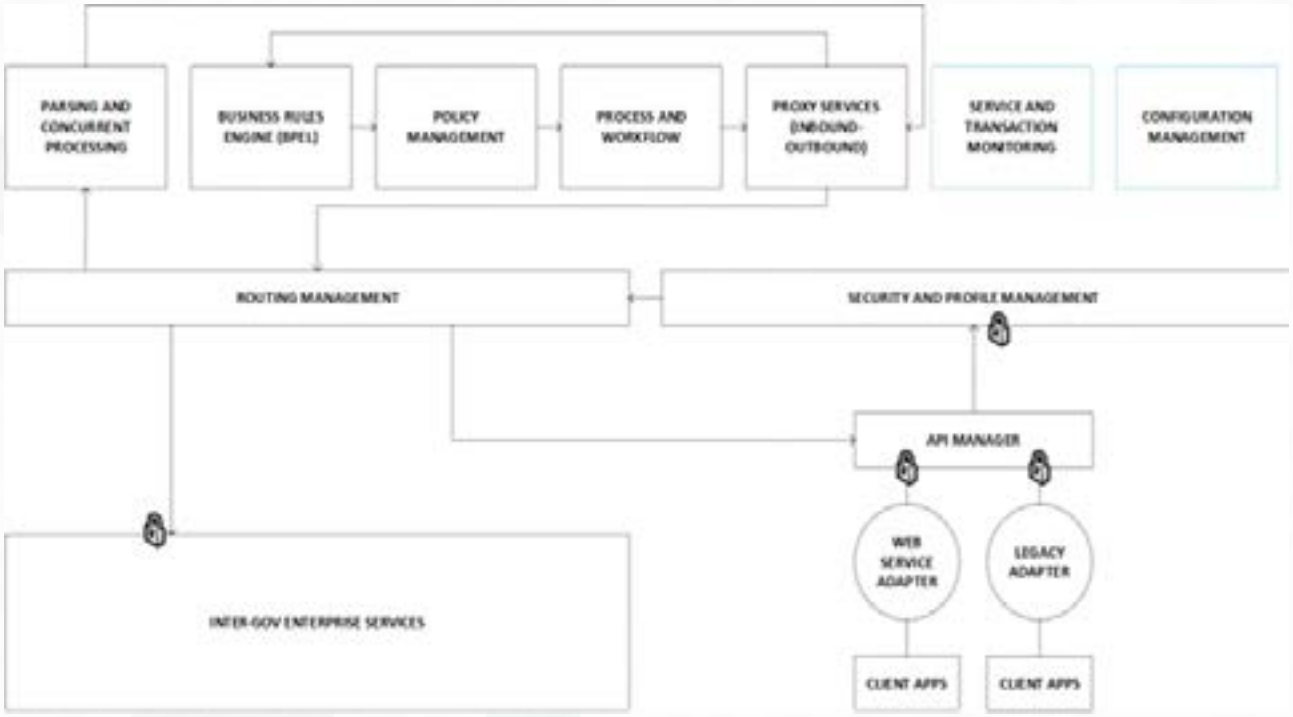


Figure 4 - Diagram of an Enterprise Service Bus

Integrated Government Platform Model

The common trap most technical architects fall into is tightly coupling applications with data sharing requirements, and vice versa. Doing so results in rigid and compartmentalized technical designs that are certain to become a future problem.
There is a fundamental distinction between reining in processes by prescribing applications to use and how each of these applications should function. So is providing a core foundation that facilitates the efficient sharing of data and information among interacting agencies regardless of the type or nature of applications.
Controlling application behavior by reining in processes is most suited for enterprise environments where the scope and enforcement of policies and business rules are governed and restricted to a single organization.
The Government agencies shall avoid Design rigidities by designing a technical environment based on a Service Oriented Architecture model.
A service oriented model, as opposed to other technical architecture models (monolithic architecture, layered architecture, component architecture), allows system architects and developers to de-couple the functional and technical components into logically defined service components by employing an enterprise service bus (Figure 4) to moderate transactional traffic and to support the efficiency, scalability, extensibility, and performance requirements of the enterprise or the government universe of meta-enterprises. By de-coupling applications, service components can independently scale and extend without affecting other components within the technical environment.

An added advantage of employing a service-oriented architecture is that, unlike any of its predecessors, So A supports agile process alignment by de-coupling governance and policy related aspects from any source environment. This gives the enterprise the flexibility it needs to adapt to the changing policy, functional, and technology environments while inoculating subsystems from effects brought about by changes to any of its component parts.

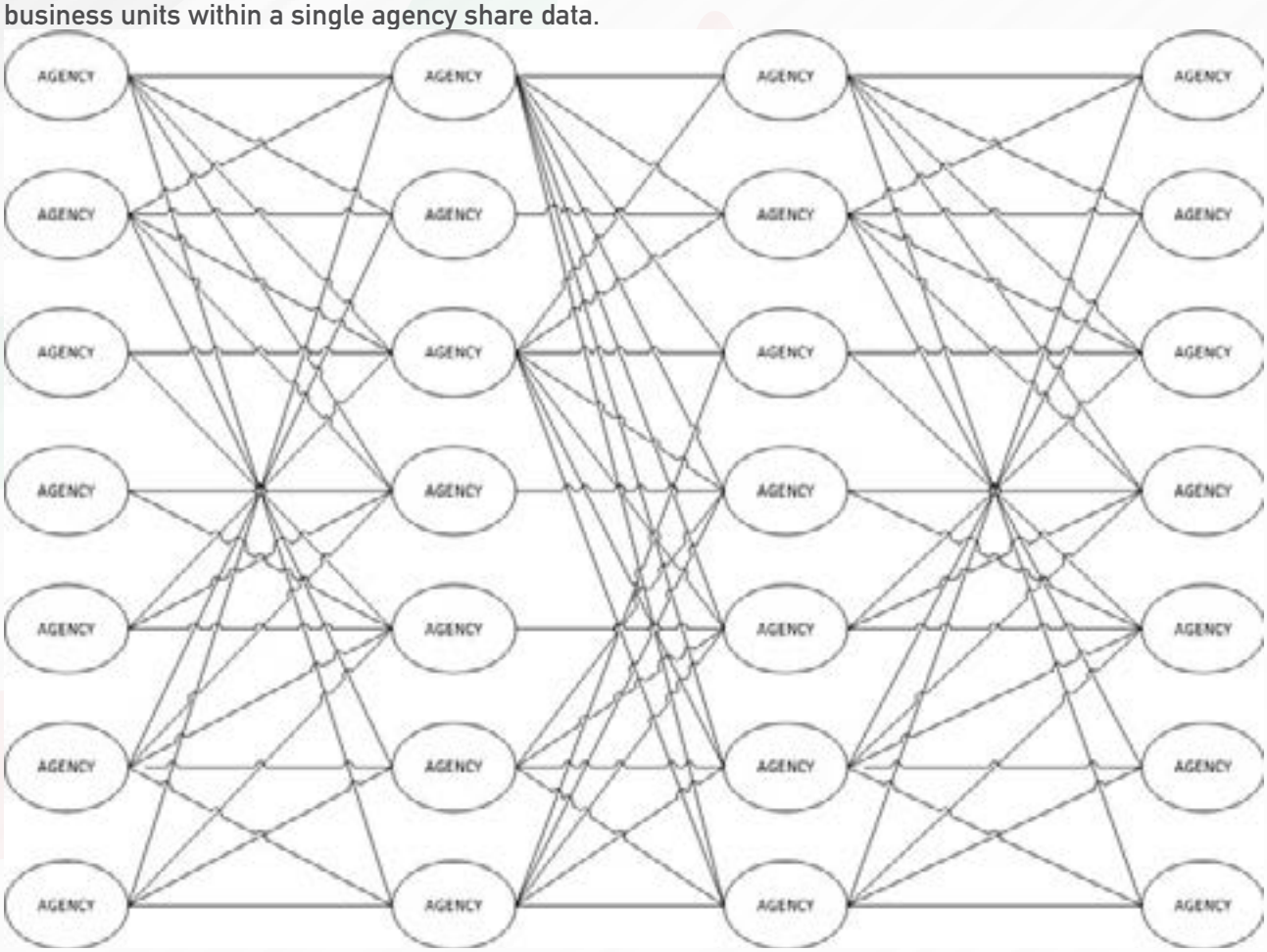
SOA is not a methodology but a design paradigm. A service or intend architecture or SOA details the "what and how" of systems design, while an implementation methodology details the "how and when" aspects of systems implementation. When referring to a service-oriented architecture, we must remember that this involves a technical design approach.
The challenge confronting the design of an environment conducive to data sharing between agencies across the whole of government is the disparity of systems employed, or the absence of systems in many cases, that need to be factored into the design of the future or target environment.

To build in the requisite flexibility, it is imperative that the design of the envisioned architecture of the future technical environment allows legacy systems as well as "bleeding edge" technologies to co-exist and interoperate. This would best be addressed by employing an Enterprise Service Bus (refertoFigure4–Diagram of an Enterprise Service Bus).

The Desired End of the Integrated Government Platform

The desired end of the Integrated Government Platform or SGP is to provide a scalable, extensible, and application agnostic platform for government agencies to efficiently share and exchange data or information with each other. The desired end is not so much to dictate or control the type of applications that the various agencies of government will use, but to ensure that essential and mission critical data and information could be efficiently shared between authorized governments agencies–regardless of source.

Figure5 illustration show government agencies currently share data. This is also applicable to how business units within a single agency share data.



Each agency "node," as illustrated in Figure 5, is managed independently. The success of data exchanges between source and target nodes in this environment heavily relies on the compatibility between technology platforms and on the management disposition to share or exchange data. This disposition either enables or inhibits data sharing and information exchange.

The current data-sharing environment of government introduces unnecessary complexity in the exchange of data. Because exchanges between sender and receiver are managed on a "per-node" basis, not only does it require additional effort and manpower (technical and functional) to maintain each node, but this current environment also negatively impacts performance and makes technical and policy-related troubleshooting or modifications difficult, complex, and highly prone to error.

The higher the occurrence of error, the higher the latency. And the higher the levels of latency, the lower the perception of users concerning the dependability toward the overall system. Furthermore, maintaining a high number of independent nodes also presents administrators with more potential points of failure, which is something they must strive to minimize.

Figure 6 illustrates the (future) structure that will be employed by the government agencies and ICTA. The structure is intended to reduce management complexity while simultaneously providing the whole of government a way to realize a more scalable, extensible, sustainable, and manageable environment that is conducive to data sharing and exchange.

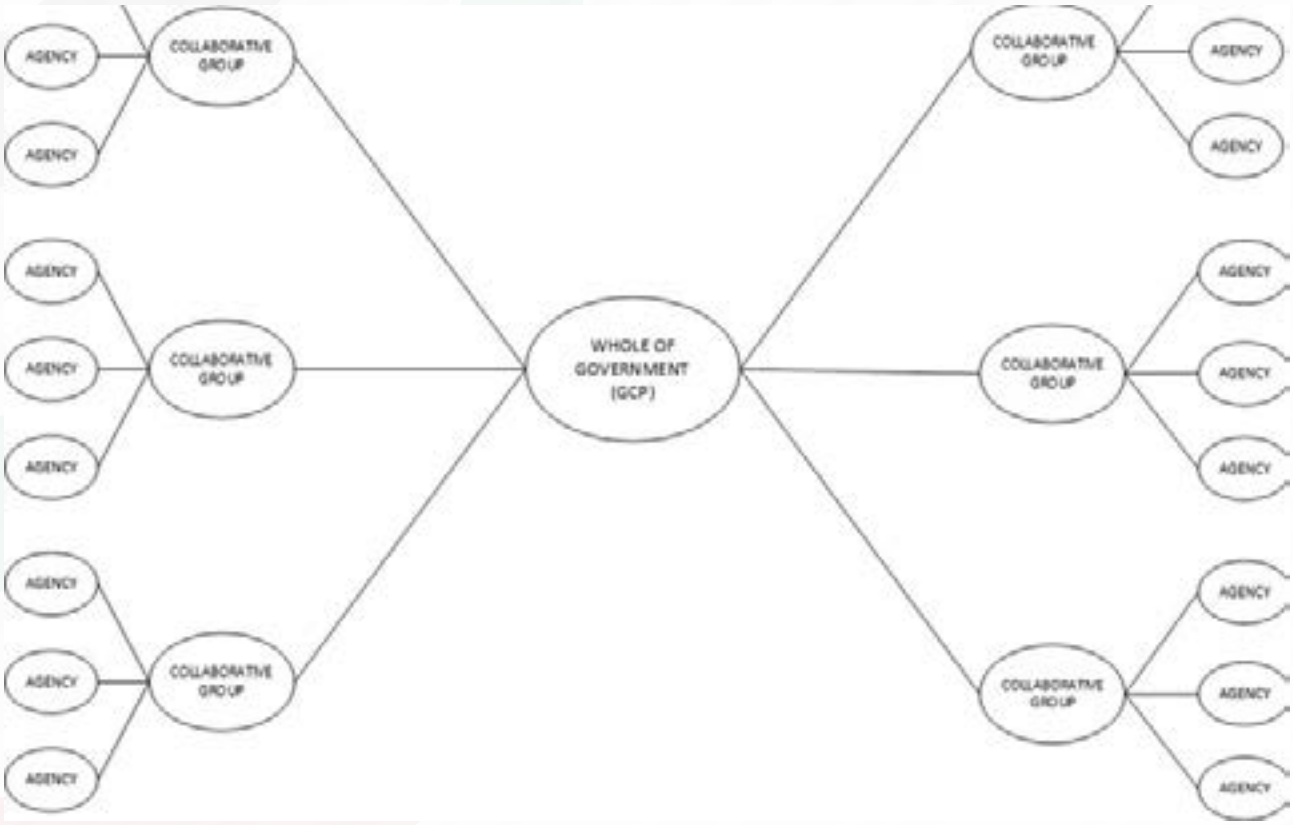
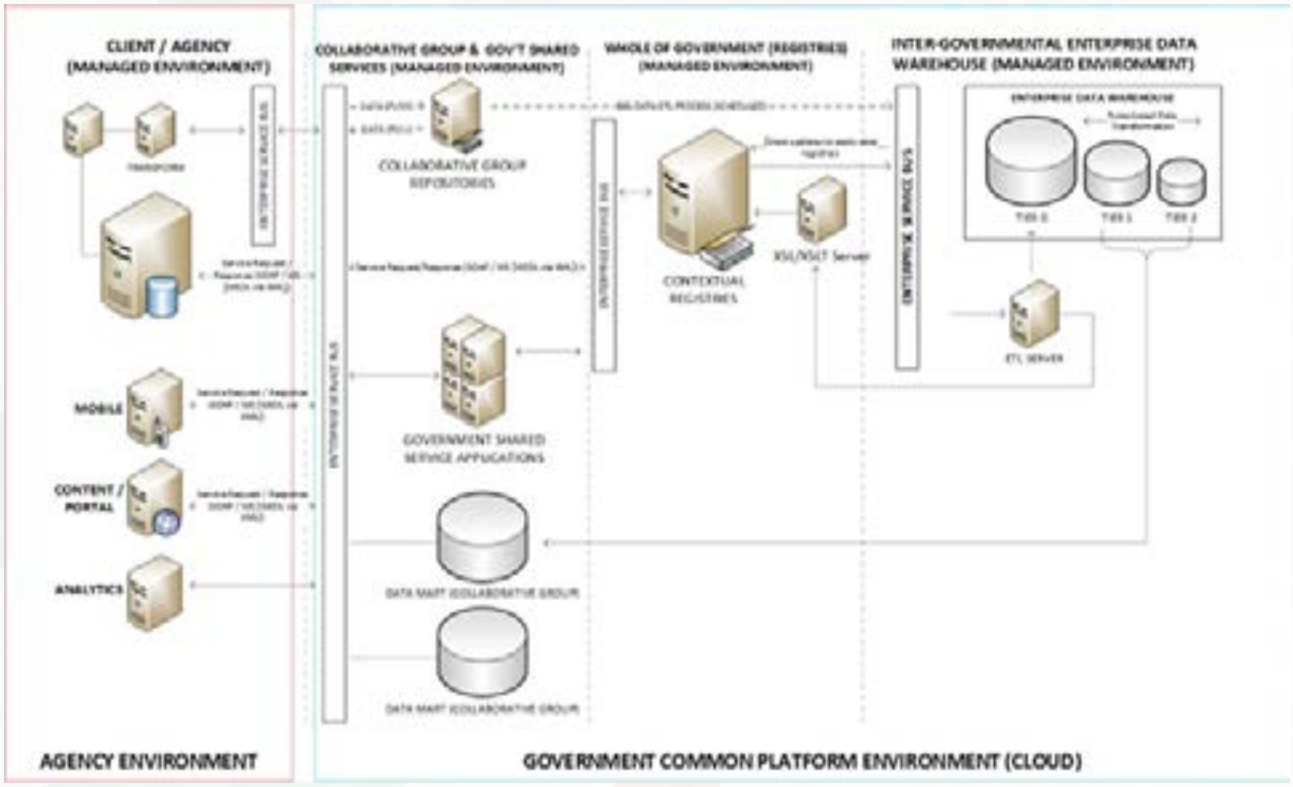


Figure 6 - Data Sharing Environment in the Government Common Platform

The scope of the interoperability framework, which is to enable data sharing and exchange for the whole of government, is immense and should be reason enough for us to structure participating agencies into federated collaborative groups that are easier to manage and would be more agile to respond to the changing demands of what data needs to be shared or exchanged between members of the collective.

The Integrated Government Platform Technology Environment and Domains

Figure 7 - Federated Technical Environment of the Government Common Platform



The Government Common Platform technology environment is organized into two primary domains: client/agency technology environment and the cloud environment. The cloud environment domain, on the other hand, is organized into three managed environments, namely: the collaborative group managed environment, the whole of government registries managed environment, and the intergovernmental enterprise data warehouse managed environment.

a. Client/Agency Managed Environment

The client/agency managed environment consists of agency-specific enterprise-grade, client/server, web applications, stand-alone applications, mobile applications, content management systems or portals applications, and business intelligence and analytics systems. These systems could either be on premise (agency data center maintained) systems or hosted within the government data centers or cloud environment.

b. Collaborative Group Managed Environment

The collaborative group managed environment will house its own enterprise service bus, MDM system, core applications, data repositories, and data marts for the exclusive use of the collaborative group and its authorized members. It will also house all government shared applications.
Applications and repositories within this environment will run exclusively on the government cloud platform. For performance requirements, each collaborative group environment will run on its own collaborative group virtual machine instance (VMI). The minimum number of VMIs housed in this managed environment will be equal to the number of collaborative groups within the SGP network. Each VMI will contain a pre- configured SGP base technology stack from persistency to presentation.

c. Whole of Government Registries Managed Environment

The whole of government registries managed environment will house its own enterprise service bus, XSL/XSLT server, and the context registries or clean data repositories accessible to authorized and authenticated users and applications. Data housed in these secured registries will serve as the official, read-only "clean" record for the consumption of the whole of government. At the minimum, this environment will house the citizens' registry, vehicle registry, business registry, and all such registries as may be required by the government.

Repositories within this environment will run exclusively on the government cloud platform. For performance requirements, each context registry environment will run on its own virtual machine instance (VMI). The minimum number of VMIs housed in this managed environment will be equal to the number of context registries within the SGP network. Each VMI will contain a pre-configured SGP base technology stack.

d. Inter-Governmental Enterprise Data Warehouse Managed Environment

The inter-governmental enterprise data warehouse managed environment will house its own enterprise service bus, extract-transform-load (ETL) server and enterprise data warehouse repositories. EDW repositories within this environment will run exclusively on the government cloud platform. To maximize performance and reduce latency, the inter-governmental enterprise data warehouse managed environment will run on multiple clustered virtual machines.

The Government Common Platform Standard Systems Environment

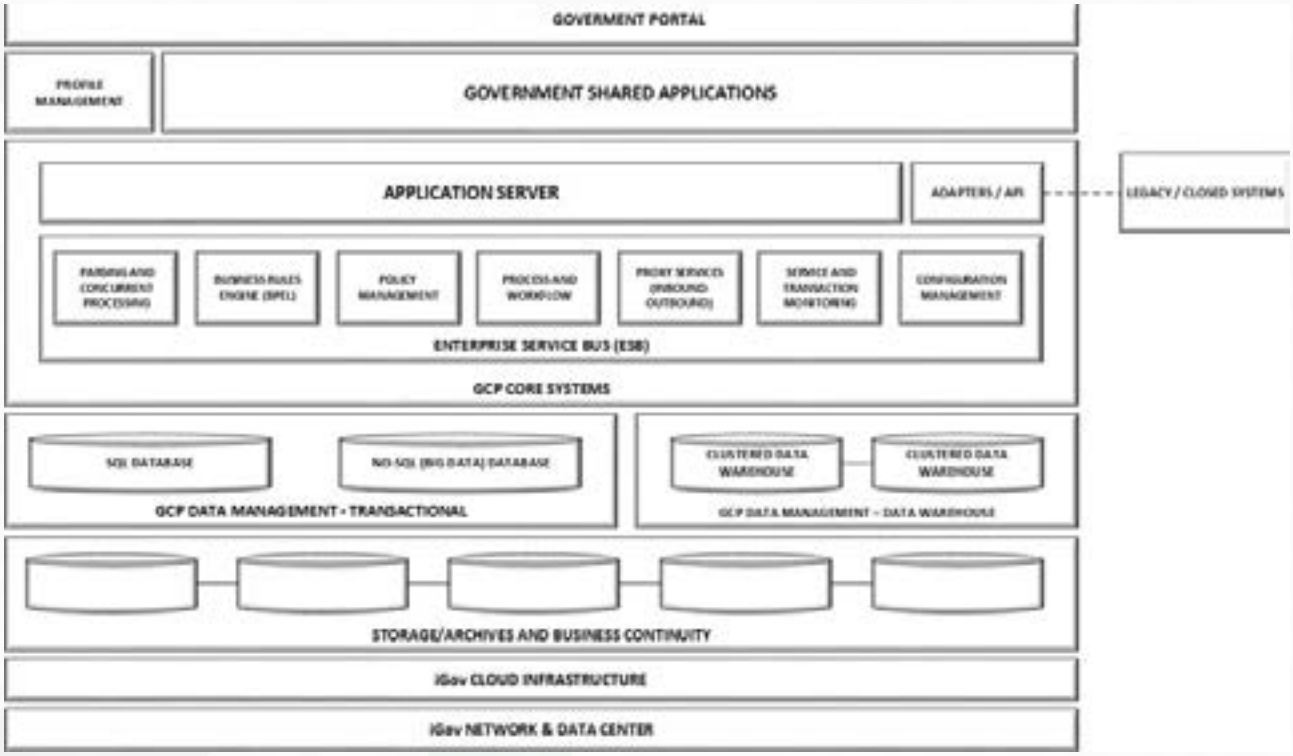


Figure 8 – Shared Government Platform Systems Environment Diagram

Figure 8 illustrates the different hardware, software, and network tiers and components that make up the systems environment of the Shared Government Platform. The design of the cloud infrastructure consisting of the network, data center, storage, and business continuity tiers are defined and managed by the Government Data Center team of the ICT Authority.

The SGP system "sits on top" of the GCCN infrastructure as illustrated in Figure8. There are four system tiers that make up the SGP systems environment: the data management tier, the core system tier, the government shared applications tier, and the unified SGP portal. The data management tier houses all static and transaction databases or repositories and the enterprise data warehouse. The core systems tier houses the enterprise service bus and the application server. The illustration (Figure8) depicting the "adapters/API" as a single component is for illustration purposes only since the regulating connections to the SGP via an adapter or API is a function integral to the enterprise service bus or ESB. The government shared applications tier will house applications that will be made available and enforced as standard for the whole of government2.

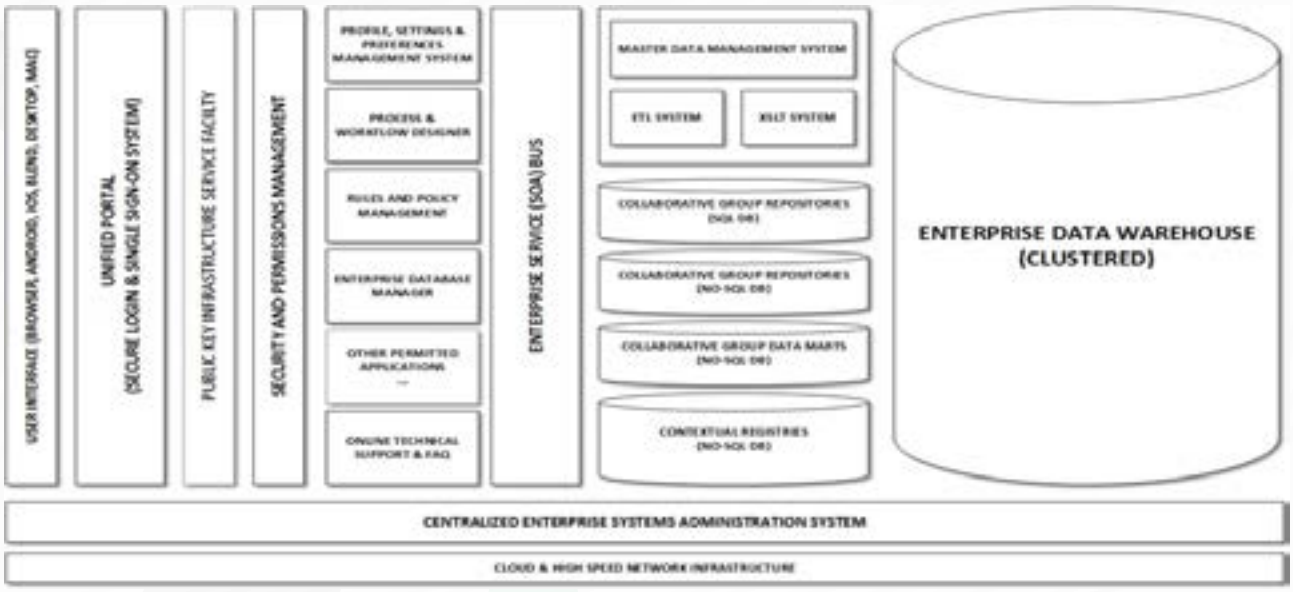


Figure 9 - Alternative View of the SGP Systems Environment

The unified SGP Portal, as indicated by its "unified" label, is the single entry point of users (functional and systems administrators) into the SGP systems environment. "Port lets" provide users access to their authorized environments, applications, and features within each application.3

Figure offers an alternative view of the components that make up the SGP systems environment organized into vertical bands. It also includes a master data management (MDM), extraction-transformation-load (ETL) system component, and XSL/XSLT component in the vertical band of data management. It must be noted, that the rules that drive the MDM will reside in the business rules engine of the ESB.

High Availability and Process Agility

Unlike other "enterprise" systems that could tolerate "downtime" as system usage is typically at its lowest on non-working days, the SGP cannot afford such a luxury–it cannot go down as a whole asset needs to be available 24x7x365.

Not only must the SGP maintain higher than normal service levels, it must also allow app- level functionality to undergo changes without having to "down" any of its component parts, and allow changes to be rolled out from staging to production at runtime.

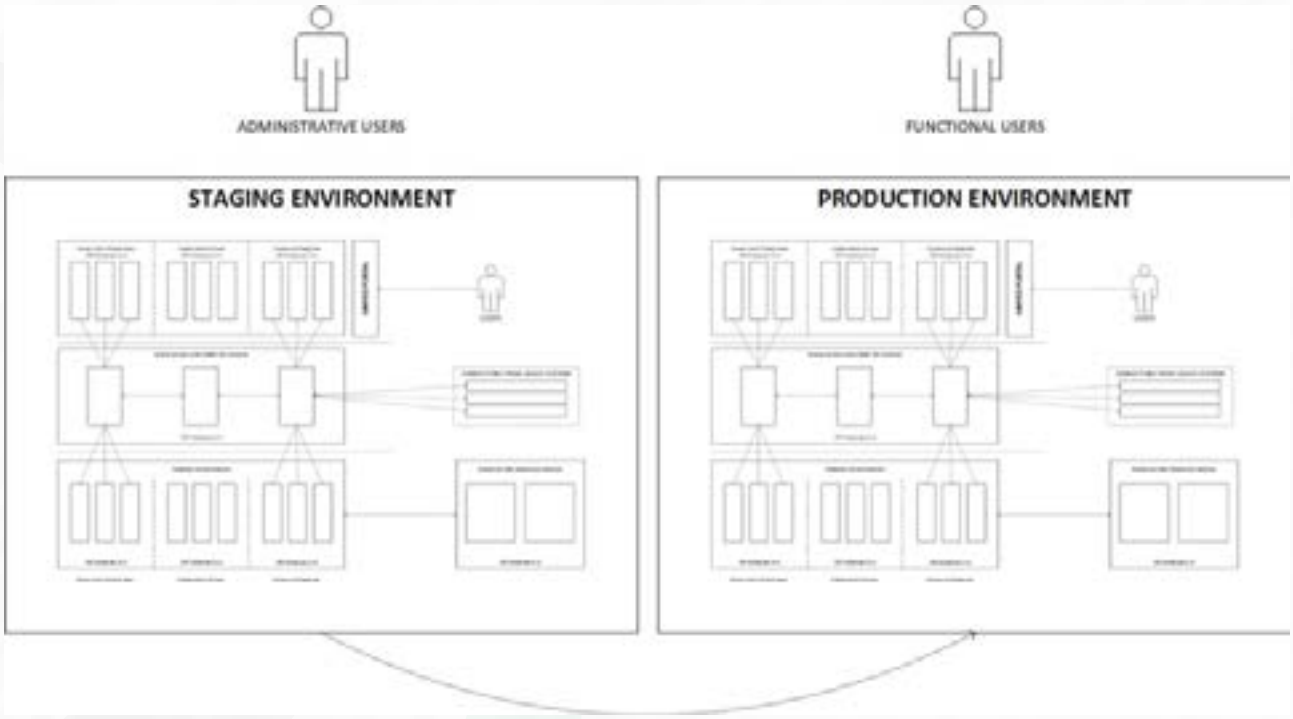


Figure 11 - Staging and Production (Controlled) Environments

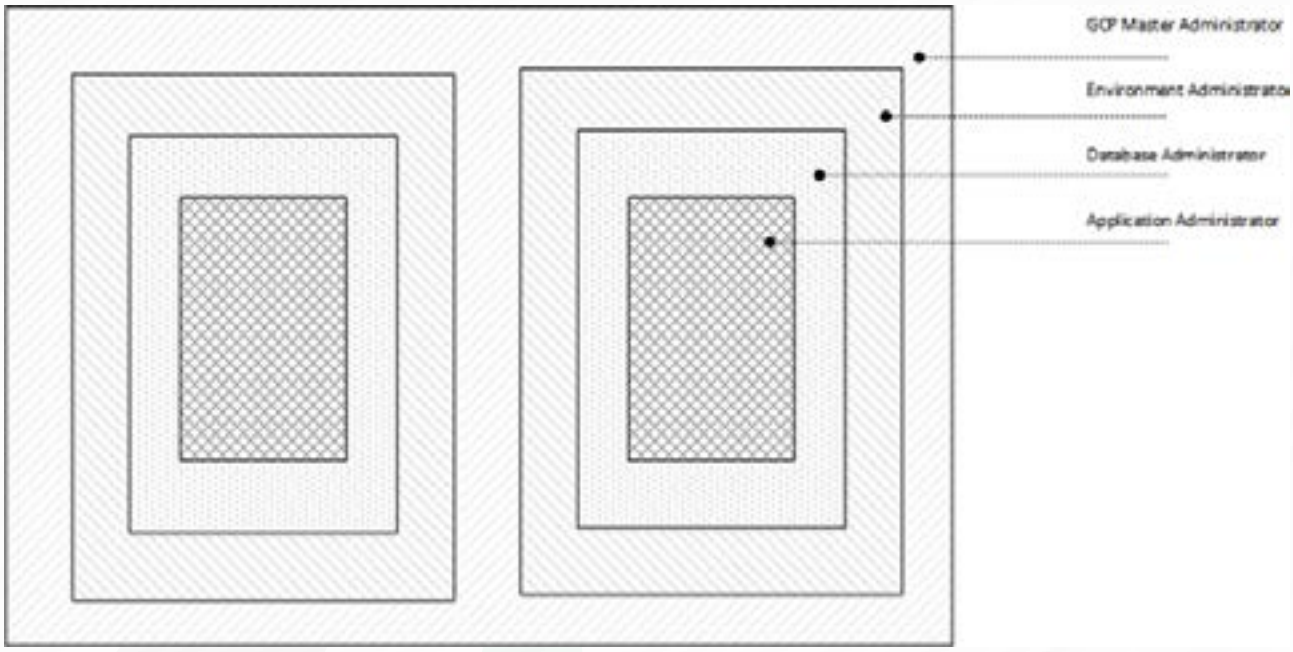
The agencies shall be required to establish different environments when the system development starts i.e development, Staging and Production (Controlled) Environments
Customization that requires modification sat the code level will only be performed in their respective development environments. Since the staging environment is an exact replica of the production environment (just smaller in size), only those applications that are ready for testing will be allowed into the staging environment.

User Access

End users (public users, administrators, and functional users) will access the SGP exclusively via the unified portal. As mentioned earlier, port lets provide users access to their respective environments for which they are authorized to use.

Security and Administration

Roles and permissions to specific instances will be centrally managed by SGP system administrators, while environment administrators will manage all roles and security permissions at the level of their environments for which they have been granted administrator privileges. The Public Key Infrastructure (PKI) subsystem will be integral to the security environment of the SGP. PKI tokens will be required to authenticate all users and their activities in areas of the SGP where a user login is required.



The centralized administration of the SGP systems environment provides for four levels of administration, each with defined administrator-level roles and privileges to enable compartmentalization essential to maintaining security levels. These levels are: SGP Master Administrator, Environment Administrator, Database Administrator, and Application Administrator. Rights and privileges would have to be made explicit to allow any of the different administrator roles access to other layers of the system.
The SGP Master Administrator, while having top-level access privileges to the environment, will not have access to lower levels by default. Environment Administrator privileges would be limited to the SGP core system components (see Figure 8). Database Administrator privileges would be limited to the database tier, while Application Administrator privileges would be limited to specific applications. Because of the sheer magnitude and scope of the SGP, no administrator privileges with far reaching authority will be granted to a single individual.

Data Sharing and Exchange Standards

Since determining how agencies will share and exchange data is the first step to creating value for the SGP, collaborative groups will be in a better position to explicitly define what data elements will be shared between interacting agencies.

This approach prioritizes the desired end of the SGP – data sharing. The more explicitly data elements for sharing are defined, the less the need for any one agency to comb through complex data structures (noise) of their internal applications just to hunt for pertinent data elements to share; and the more valuable the SGP network of collaborative groups becomes to the whole of government.
The SGP's desired end is also to ensure that the quality of data shared via the SGP and agencies is auditable, credible, and up-to-date.

Data exchanges between interacting agencies within a collective will be in either structured (SQL-based) or non-structured (non-SQL-based) formats, or both, depending on the need and technical context. Building in there quested flexibility to enable members of a collective to share both structured and non-structured data not only allows backward compatibility, but more importantly, also sets the foundation for a future-proof, extensible environment.

Shared applications accessible via the SGP are grouped into two environment categories: Personnel self-service applications and shared enterprise applications.
The category of applications domiciled within the SGP are intended to address information requirements at both the individual and organizational level; thus, creating a closed loop environment that captures data at the atomic (personnel) level of the organization and allows more efficient cross-referencing between actual behaviour and pro- forma processes.

Personnel Self-Service Application Environment

These consist of a set of applications for government employees to access standard productivity tools (such as Email and Calendar), view and update personal information

(Such as Payroll and Benefits), transact with other departments within their organization, and perform individual transactions directly with other government agencies (such as the GSIS, BIR, CSC, PHIC, etc.) through SGP application or out lets.

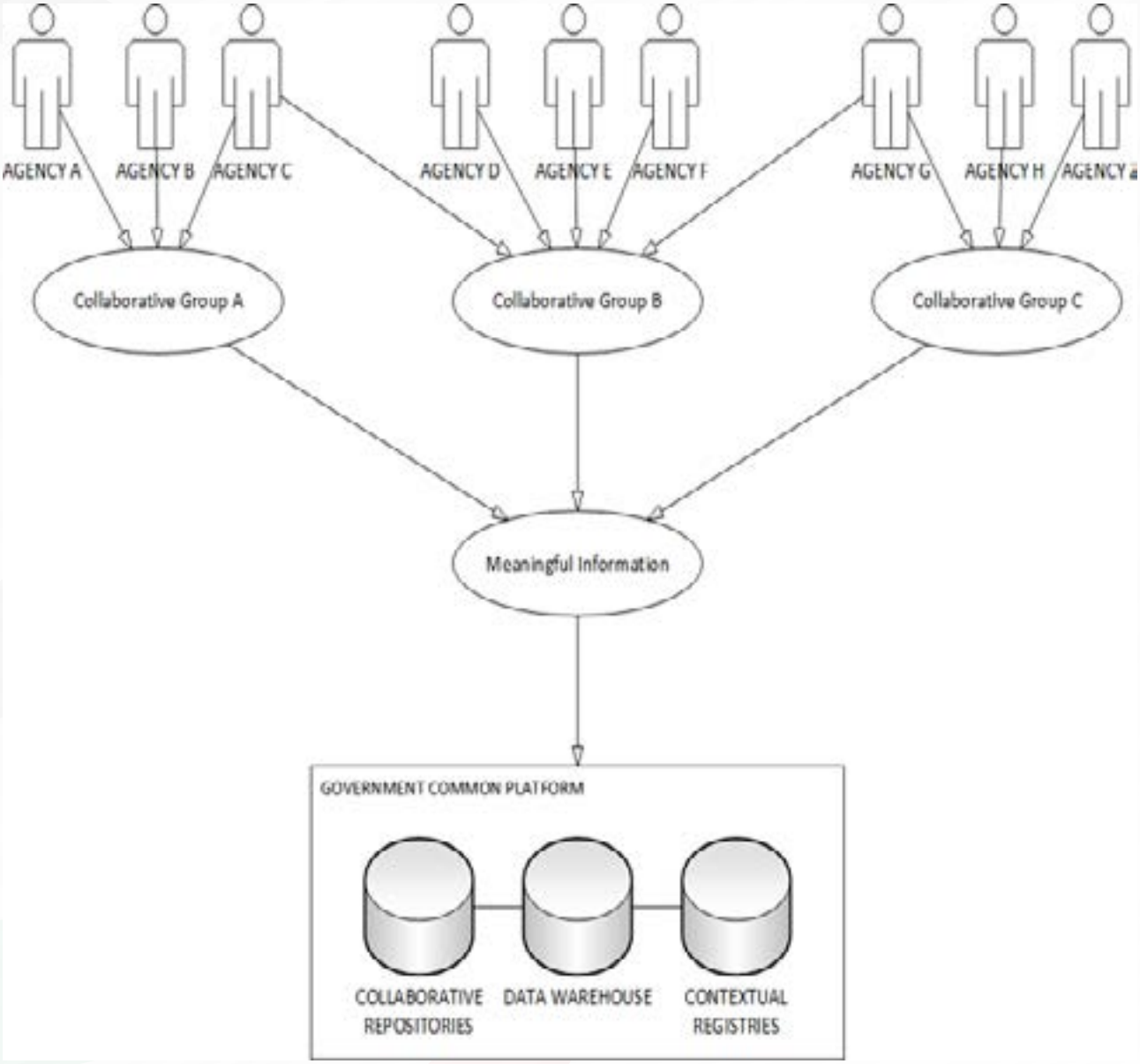
Shared Enterprise Applications

These consist of a set of applications guided by a common government standard (such as those involving accounting, budget management, finance, treasury, etc.) and applications commonly used by government agencies such as contract management, project management, and document approval routing. It will also provide content port lets that will house government content intended for universal consumption. We cannot overemphasize the importance of producing meaningful information through the SGP network.

Meaningfulinformationentailsprovidingnearaccurateandup-to-dateinformation, using data sourced from agencies (or in this case, from the repository of collaborative groups) having the closest proximity to the transactions or activities of a person.
If the SGP could allow members of collaborative groups to have access to relevant information on-demand, then the benefits derived from achieving this level of efficiency will produce a disproportionately higher value. This higher value will rationalize the required financial investments to establish, manage, and sustain such a collaborative environment.

Value Flows: From Raw Data to Meaningful Information

Figure 18 - Generating Meaningful Information through Collaborative Groups



Minimum Standards for the Interoperability Technical Architecture

Section Overview

This section details the technical architecture and corresponding technology environment and specifications of the Integrated Government Platform. The scope of the SGP is to enable data sharing and exchange for the "whole of government."

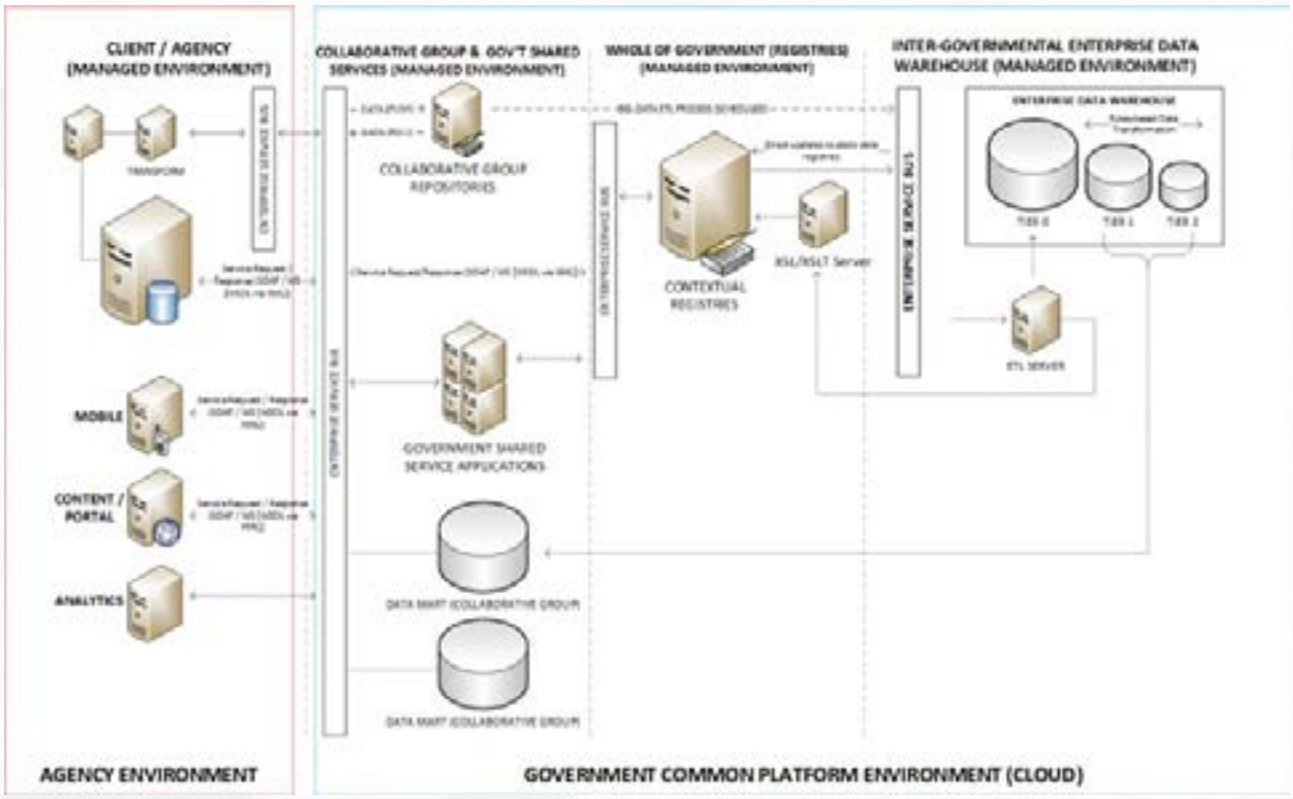


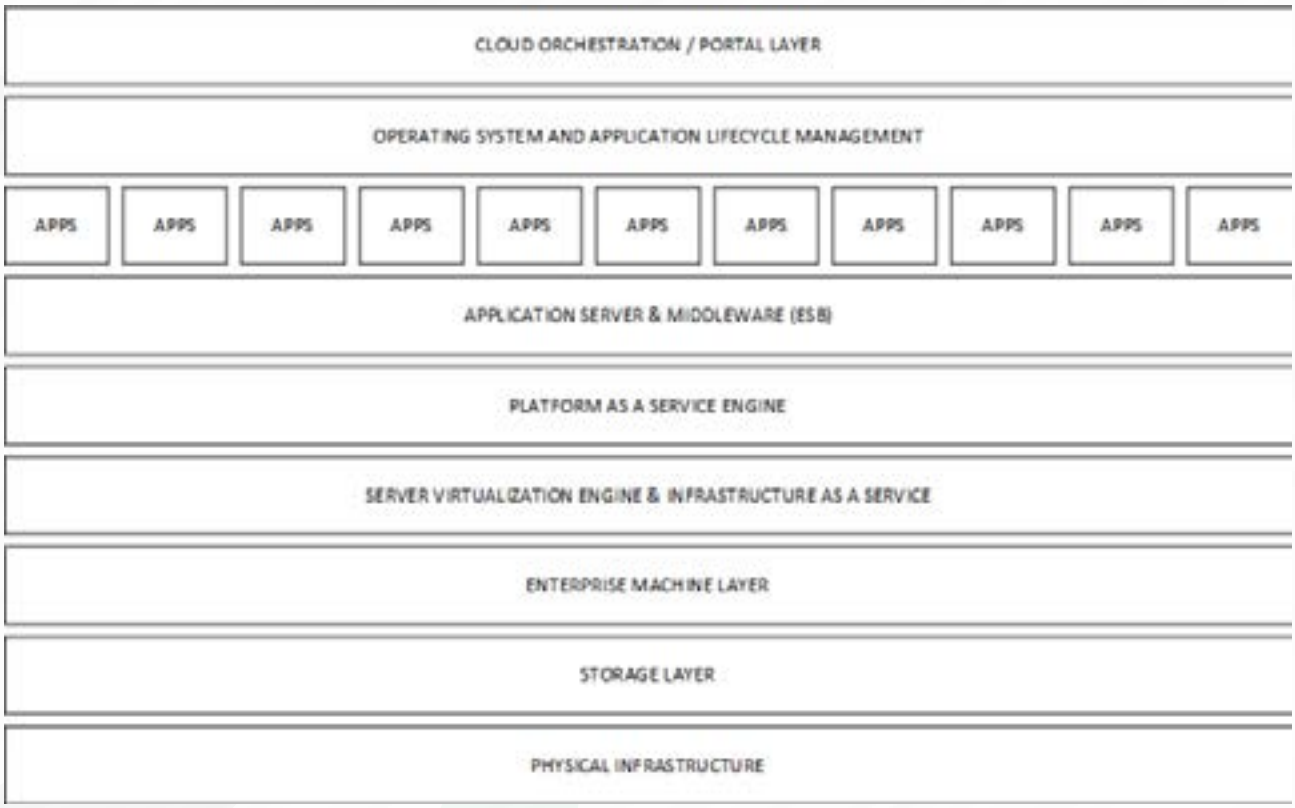
Figure 9 - SGP Technical Environment

General Technology Environment Specifications

The technology services of the government will be tiered into the following offerings:

- Infrastructure as a Service(IaaS)–the Cloud Infrastructure
- Platform as a Service (PaaS)–the Government Common Platform
- Software as a Service(SaaS)–the Government Shared Applications
- Database as a Service(DBaaS)–the enterprise no-SQL database system

Thissectiondetailsthevariouscomponentsessentialtodeployingthe SGP's "Platformasa Service" offering (see Figure).



The general technology environment specifications detailed in this section will apply to the development, staging, and production environments of the SGP and agencies that are grouped together to share platforms and systems. The technology environment specifications are as follows:

1. The SGP shall be hosted on Government Cloud (GovCloud) virtual servers.

2. The SGP shall be structured in to three managed domains:

2.1. Collaborative Group and Government Shared Services Domain
2.2. Contextual Registry Domain
2.3. Inter-governmental Data Warehouse Domain

3. The SGP architecture shall be service oriented and shall be tiered into the following de-coupled environments:

3.1. Orchestration Environment
3.2. Enterprise Service Bus/Middleware Environment
3.3. Data Management Environment

4. The SGP environment shall be capable of supporting a minimum of 50,000 concurrent transactions per minute.

5. The SGP shall maintain a maximum latency of 1.5 seconds for search; maintain a maximum latency of 3.0 seconds for transactions; and, shall maintain a base service uptime of not less than 95.9999% SLA per year.

6. The SGP Orchestration Environment shall consist of:

6.1. Enterprise Portal System
6.2. Enterprise Web Server
6.3. Identity Management System
6.4. Enterprise Application Server
6.5. Enterprise Business Intelligence and Analytics System

7. The SGP enterprise service bus environment shall consist of:

7.1. Enterprise Middleware System
7.2. Business Rules Management
7.3. Automated Policy and Workflow Management
7.4. API Management System (legacy and WSDL)
7.5. Parsing and Concurrent Processing System
7.6. Proxy and Routing Services

8. The SGP data management environment shall consist of:

8.1. Enterprise Data Management
8.2. Big Data Management System
8.3. Enterprise Master Data Management Server
8.4. Extract-Transform-Load (ETL)System
8.5. Extensible Style sheet Language Template(XSLT)System
8.6. Enterprise No-SQL Database Management System
8.7. Enterprise Relational Database Management System
8.8. Enterprise Data Warehousing/Data Mart System

9. The SGP application development environment shall consist of:

9.1. Capability to support runtime applications in the following development environments: PHP, Java, and.NET.
9.2. Standard integrated environment widgets to best-of-breed Integrated Development Environment or IDE tools.
9.3. Standard integrated environment widgets to best-of-breed Software Configuration Management or SCM tools.
9.4. Standard integrated environment widgets to best-of-breed Software Build Management tools.
9.5. Standardintegratedenvironmentwidgetstobest-of-breedSoftwareTestingtools.
9.6. Standard integrated environment widgets to best-of-breed Software Planning tools such as Scrum and Open Unified Process, and must support agile or tradition software development lifecycle management processes.

SGP Core Application Environment (Orchestration Tier) Specifications

The SGP core application environment (orchestration tier) is a complete and base- configured GovCloud-resident platform of integrated standard applications where all next generation applications that serve the whole of government will be deployed, consisting of an Enterprise Portal, an Enterprise Web Server, an Identity Management System, an Enterprise Application Server, and an Enterprise Business Intelligence and Analytics System.

Enterprise Portal System

Theenterpriseportalsystemwillserveasthestandardentrypointintothe SGPapplication environmentthroughtheuseofportletsforadministrationfunctionssuchasapplication management, database management, identity and security permissions management, and virtual machine instance management, and transactional functions such as those involving the use of government shared applications.13 the enterprise portal system will provide a foundation for homogeneity in user experience.

The base specifications of the enterprise portal system are as follows:

• Security Services Must support single sign-on and standard security certification (X.509, SSL, OCSP/CRL, and PKCS12) protocols
• Standards Compliance Must be JSR 168/286, WSRP 1.0/2.0 compliant
• Integrated Systems Management Must support integration services to the SGP application environment, business rules, events, data, and government shared applications.
• Portal Management Must provide UI-based integrated functionality to control all portal components, user transactions, provide access to log files, user permissions, and so forth.
• Port lets Must provide integrated environment to develop, configure, integrate, and deploy personalizedend-userOLTPandOLAPapplications.Portlets,attheminimum,mustbe able to efficiently render user interfaces developed using standard and semantic web specifications (inclusive of geospatial rendering standards). They must also be WSRP 1.0/2.0compliant.

• Policy Management a. Must be able to govern SOA interactions through security and operational policy management and enforcement. b. Must support WS-Security 1.0, WS-Policy Attachment1.0, WS-Security: Username Token c. Profile1.0, WS-Security: X.509TokenProfile1.0, WS-Security: SAMLTokenProfile1.0, WS-Security Policy, SAML1.1. d. Must support configurable mechanism to apply or remove a policy from a service. e. Must support custom policy.
• Integration Services Must provide integrated UI-based management environment to manage integration to and from the ESB.
• Reporting Must provide UI-based functionality for users to perform user-defined and standard data extracts for both standard and adhoc reports.
• Collaboration Services Must provide UI-based functionality for users to define, design, manage, route, and track workflows.
• Unified Communication • Must provide UI-based functionality for secured online and GSM-based messaging, electronic mail, IP-based video conferencing, calendar and scheduling, task management, and workflow management. • Must also provide social networking functionality.
• Content Management • Must provide UI-based functionality for users to perform content authoring, collaborative review, content syndication, store, index, retrieve, organize and classify, and manipulate multimedia objects in both structured and non-structured formats.
• Basic and Advanced Search • Must provide UI-based functionality for users to perform low latency basic search and parameter-based searches against structured and non-structured data repositories.
• Uninterrupted Runtime Deployment of Application Customizations and Modifications • Must provide UI-based functionality for users to customize and deploy modifications to user interfaces, navigation flows, and content at runtime without the need to impose a mandatory server stop-start to restart the environment.

Enterprise Application Server Base Specifications

The main service function of the enterprise application server is to provide the essential abstraction to de-couple the presentation (user interface) and data tiers from the application tier regardless of the type or nature of an application.
Given the scope of the SGP, it will need to maintain a heterogeneous environment capable of servicing and supporting standard Java, PHP, .NET and other platform independent environments inclusive of Open Source systems.

By design, the SGP will maintain Virtual Machine (VM) instances of base-configured enterprise application servers with integration adapters to:

1. SGP enterprise service bus.
2. Enterprise database repositories.
3. Standard connectors to the Java, PHP, or Microsoft.NET platforms and enterprise services.

The base specifications for the SGP Enterprise Application Server are as follows:

Features and Core Services

- Virtualized Environment Optimization.
- Must have high-availability clustering functions.
- Integrated Web Server services.
- Micro services support.
- Application Logic and Intelligent Routing.
- Garbage Collection.
- Comprehensive Technology Foundation Library.
- XML-based Application Configuration File Management.
- Mustsupportthefollowingopenstandardsandspecifications:JSP2.1,JSF1.2, Servlet2.5,EJB3.0,JAX-WS2.0,JMS1.1,JNDI1.2,JCA1.5,JTA1.1,JACC
- And JAAS 1.0, JMX1.2, J2EEApplicationDeployment1.2, J2EEManagement1.1, JDBC3.0, JAX-RPC, JAX-WS, Enterprise-class JMS, JPA, POJOs.
- Must have natives support for REST, SOAP, UDDI,WSDL,WSRP,WS-Security.
- Must have Transaction Logging (TCP/UDP).
- Dynamic Clustering and High-Availability.
- Dynamic Fail-over Support.
- Dynamic Load Balancing Support.
- Standard Library of Database Connections (RDBMS, NoSQL) APIs.
- Managed Application Deployment and Migration Environment.
- Web Services Configuration and Deployment.
- OLTP Batch Processing Configuration and Management.
- Web2.0andModularApplicationDevelopmentSupport.
- Out-of-the-box support for JavaEE, Microsoft.NET, and PHP applications.
- Automatic Resource Management.
- Cross Component Tracing.
- Integrated Development Tooling Support.
- Centralized Administration.
- Dynamic Memory Allocation (JIT-usage) and Termination.
- Dynamic Application State Roll-back Capabilities.

SGP Enterprise Middleware (Service Bus) Environment Specifications
The base specifications of the SGP Enterprise Middleware (enterprise service bus) Server are as follows:

Basic Functionality

- Distributed application management.
- Multi-OS support.
- Supports automatic workload balancing.
- Ability to support a wide range of protocols (namely HTTP, JMS, FTP, REST, File, WS-RM, MQ, SMTP, EJB, JCA, etc.) to provide a range of capabilities for newly developed business needs and to support integration with a wide range of third- party and legacy systems and services.
- Ability to accept a request in one protocol and forward it as a request using a different protocol.
- Ability to translate data from one format to another, possibly using that data to enrich data streams and make routing decisions along the way.
- Ability to connect multiple services together into a larger composite service, and manage the flow of control and information among the component services.
- Provide easy to use graphical editors for ESB flows using both standalone IDE and web-browser.
- Support file formats like XML, non-XML and binary.
- Support Transport Security, Message Security, Console Security, and Policy-based Security. Must support built-in security components and plug-in third party components.
- Provide uniform mechanisms for identifying, managing, and monitoring both technical and business errors, with the ability to customize specific error behavior as needed.

Data Middleware

- Support for Remote File Systems
- Support for Network File Systems
- Support for ODBC
- Support for JDBC
- Support for XDBC

System Programming Interfaces

- Support for Message Oriented Middleware
- Support for Java Message Service
- Support for Web Services Simple Object Access Protocol
- Support for Distributed Computing Environment
- Support for FTP, HTTP, S-HTTP, IP, SMTP, TCP
- Support for ebXML messaging, WSDL, UDDI
- Support for Service Oriented Architecture (SOA) and event-driven architecture (EDA)

Platform Middleware

- Support for fourth-generation languages (4GL) and programmable Web Servers and Micro services Web Servers.
- Support for personalization, multi-channel access.
- Support for content management.

Integration Middleware

- Support for thick encapsulation Adapters.
- Support for thin encapsulation Adapters.
- Support for proprietary and open standard technical adapters and application adapters.
- Support for device and end-point physical devices.
- Able to control throttling and load balancing to meet SLAs on a per-endpoint basis.
- Ability to cache the static data to improve performance by configuration only.
- Ability to support multiple communications paradigms such as Request/response, Synchronous and asynchronous, One-to-many, many-to-one, Publish-Subscribe, Mix-and-match (e.g. sync-to-a sync), and split-joins.
- Ability to support Advanced Service Pooling such as Routing to active endpoints and service load balancing.
- Ability to support entire Service Lifecycle Management and Governance involving Metadata Repository & Service Registry.

Data Navigation

- Native support for SQL and XQuery

Business Process Management

- Integrated graphical process design tool.
- Native runtime execution engine.
- Transaction routing, monitoring and logging.
- Post-completion analysis capabilities.
- The proposed solution must be able to handle the various flavors of the real world business processes within the same unified BPM engine, i.e., system-centric, human- centric, document centric, decision centric, social centric and analytics centric processes.
- The proposed product must be based on open standards and not using proprietary languages or scripting to develop or deploy processes.
- Ability to support advanced routing logic within work processes, including routing based on business rules, routing based on LDAP.
- Hierarchy (e.g., routing to a person and three levels up to his managers), routing based on Voting Outcome percentage. This must be achieved through the software's configuration dialog, not through custom coding.
- Provide a visual representation for viewing, creating, and editing processes via both IDE and web-based modeling tool. The models generated should be "what you model is what you execute (WYMIWYE)."
- Provide native support to execute BPMN 2.0, BPEL 1.1 and BPEL 2.0 processes within the same unified engine.
- Ability to allow web-based modeling, service configuration, deployment and testing using

web-based console.

- Easy-to-use front-end and integration with portal to provide unified user experience and role-based UI.
- Should support collaboration capability with the enterprise portal.
- Ability to easily create new process flows or modify existing flows as business needs dictate.
- Provide packaged "templates" of common workflow processes that can be used as a starting point.
- Ability to support repeatable process.
- Ability to support sub-process.
- Ability to integrate with the document repository for online document access.
- Ability to serve as basis for fully automated processing (i.e. with no human participation).
- The proposed product should provide an OOTB work list portal for process and workforce management.
- Ability to design serial, parallel, and complex processes.
- Version control on flow definitions.
- Ability to escalate work if not addressed within specific timeframes.
- Ability to generate notifications (such as email, SMS, voice, IM) for high priority, escalations, pending and overdue work items.
- The proposed product should provide API / web service that allows other applications to access work list.
- Ability to queue and reroute work based on resource availability. Round-robin assignment for a group of users should be provided OOTB.
- Ability to set reminders, deadlines and delegation.
- Built-in advanced human patterns such as escalation, re-assignment, withdraw, suspend, claim and resume.
- OOTB ability for users to specify their Business calendars which allows expiration and escalation times specified on activities to be measured in "business hours"
- Rather than arbitrary 'wall clock' time which may produce incorrect results around holidays and weekends.
- Ability for users to specify delegates to complete their work tasks when they are unavailable.
- Ability to share the comments and attachments with users participated in the same work process instance.
- Ability to allow users to view the history of the process instance that he / she is working on.
- Ability to allow users to view audit trail of the process instance that he / she is working on.
- Ability to perform simulation in desktop application and web-based console.
- Ability to do documentation of processes / activity and generate as HTML / XML.
- Ability to integrate with other systems to access work tasks (e.g. email inbox, Line- of-business, applications, etc.).
- Ability to allow process owner / administrator to alter the flow of running process instances;
- Provides a visual guide on the outstanding tasks to complete in order to accomplish the work. Milestones can be defined and setup along the process. This feature is commonly known as "Activity Guide."
- Provide "Player" feature in web-based console, which allows user to run and step through the business process in order validate the business flow, the business rules, and the associated user interfaces. This feature must provide a visual audit of the process flow during the testing.

- Ability to allow business users to easily create web form (rich, dynamic, user interfaces) for their business processes by dragging and dropping controls from the UI palette in web-based console. The fields in the web form will then be auto- generated into business objects for workflow execution.
- Provide "Adaptive Case Management" feature where enable organization to handle unstructured, ad-hoc processes and their contents and information.
- Support for importing from Visio and XPDL that is a built-in feature of the software, and fully supported by the software annual maintenance.
- The business process must be able to generate a Web service interface using in-built configuration dialog to be invoked by external applications.
- The business process must be able to generate a message interface (JMS) using in- built configuration dialog to be invoked by external applications sending a JMS message.
- The business process must be able to be configured using in-built configuration dialog to be started by timer schedule including: specific time of the day (e.g., 8 AM), specific days in the week (e.g. Every Monday), and specific days in the month.
- The platform must be scalable and provide high availability and performance.
- The platform should be easily able to easily scale appliances such as engineered systems and derive the out-of-the-box performance benefits.
- The proposed solution must include a data management system for application objects that are shared across multiple servers, require low response time, very high throughput, predictable scalability, continuous availability and information reliability.
- The proposed solution must include easy to configure HTTP session management module dedicated to managing session state in the clustered environments.
- The proposed software must have public website with samples, tutorials provided by the product team and member of the public.
- The proposed software must provide official documentations (Installation Guide, User Guide, Developer Guide, Administration Guide, Release Notes, etc.) in HTML and PDF format freely available to the general public on public website.

Business Rules Engine

- Native UI-based Business Rules Management.
- Logic Complexity Management.
- Provide OOTB dashboards and custom dashboards that can be created by business users without IT intervention.
- Should have built-in adapter to propagate the runtime business KPI from BPM engine to the real-time Dashboards.
- Provide easy to use web based editor for creating and editing rules. These tools must be part of the unified design-time environment.
- Ability to allow business users to change the business rules. The changes should be able to take effect immediately.

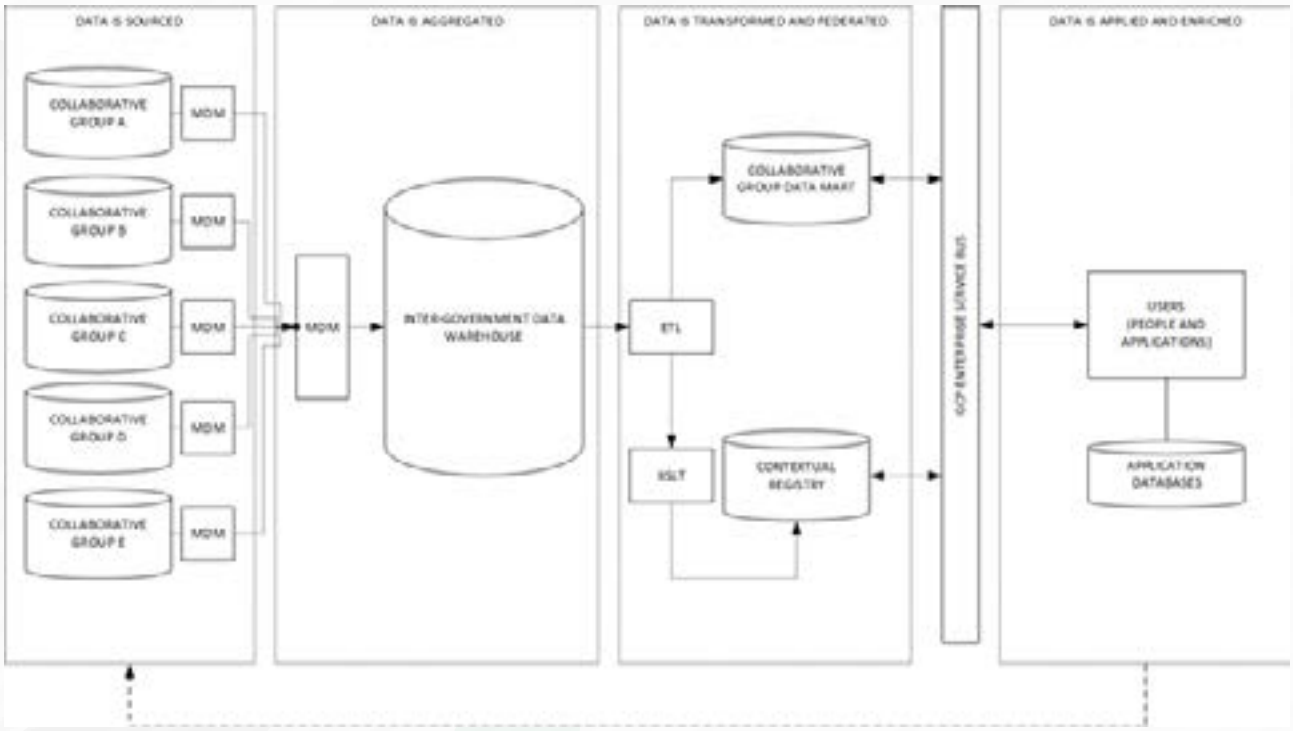
Business Activity Monitoring

- OOTB support real-time, broad and deep, multi-source and multi-channel event monitoring.

OOTB support multi-source and multi-channel event analytics.

Data Management Environment (Big Data and Data Warehousing) Specifications

Figure 32 - SGP Data Management Environment



Master Data Management System Features and Core Services

- Native integrated graphical development tool set.
- Native tool-based multi-source identification and connection.
- Supports rules-based administration.
- Supports error detection and correction.
- Providestools-basedmasterdatadesignfunctionsfordataqualitymanagement, classification, taxonomy design, item master creation, schema mapping, product codification, data enrichment and governance.
- Supports automated and rules-driven extraction, transformation, and loading (ETL) processes of transactional and big data elements to the inter-government enterprise data warehouse.15
- Supports automated and rules-based XSLT transformation.
- Supports Big Data processing and transformation functions.

Data Warehousing System Features and Core Services

- Provides native support for universal access
- Provides native metadata management functions
- Provides native data structure optimization functions
- Provides native real-time integration functions for on premise or cloud based data objects
- Provides native integrated data quality management functions;

ANNEX 8: SYSTEM INTEGRATION STANDARDS PURPOSE

A standard includes specific low-level mandatory controls that help enforce and support a policy. The purpose of this document is to further elaborate the system integration requirements which forms a critical pillar of the Systems and Applications Standard. These requirements are mandatory and must be adhered to by all members of project teams involved in development and implementation of enterprise applications, including employees, consultants and / or contractors involved in the development or modification of integrations that support the MCDA at an enterprise level.

The scope of this standard extends to and includes integrations with in-house enterprise applications as well as commercial off the shelf enterprise systems.

Requirement Level

The wording conventions below are followed in this document

Term	Meaning
Must	This requirement is mandatory, it is not optional.
May	If there are options provided, the implementer can choose one or more of the options outlined. At least 1 option must be selected.
Should	If business rules contradict a standard practice, deviating from the standard must be approved by management as a modification to the standard practice

Application

Computer programs, procedures, rules and associated documentation and data pertaining to the operation of a computer system.

General Integration

- All integrations must follow the requirements outlined in this document
- Objectives for system integration are:
  - Security of data in transit and at rest
  - Simplicity of use
  - Ruggedness (difficult to misuse, failsafe when errors are encountered)
  - Reliability
  - Efficiency i.e. fast enough for the purpose it was created, either real-time (synchronous) vs batch or regular updates (asynchronous)
  - Minimum development cost
  - Conform to standards
  - Ensure data is sent to/received from trusted endpoints. Data integrity must be maintained
  - Define user needs, roles and permissions (no back-door access by users)
  - Enable non-repudiation. If system A sends a message to System B and it fails:
  - Retry the request x number of times
  - Display an error message to the user if possible
  - Send and/or log an error message to support team

- Standard naming conventions
- Consider amount of information being integrated to determine method of integration (i.e. will integration be one record on demand, or hundreds of records each time integration is run).

API Key Management

- Use one private key per application, or for multiple applications use one key for applications with similar purposes
- Name keys according to use cases
- Make the API keys read-only if the service should not be updating information
- Deliverables:
  Document with appropriate key information, that is secured

All passwords and keys (security tokens, etc.) are to be stored securely.

Security

- Transferring confidential data through non-secure mediums (e.g., unofficial email, USB, inter-office mail (paper) is prohibited. If absolutely necessary, files must, at a minimum, be password protected.
- Ensure confidentiality forms exist when dealing with 3rd party companies that work with Government data. The recipient/parties involved need to be aware of the sensitivity of the data and ensure the data is protected at all times and destroyed as required
- Encryption in transit and at rest
- No re-purposing of user accounts

Strong passwords

Preferred file format

- XML
- Excel (tab, comma, etc. delimiters)

Fixed field text (not preferred but acceptable if other formats not available)

Preferred Integration Methods

- Selection of method should take into consideration performance of integration
  Completion in timely manner
  Real-time vs scheduled task & frequency of execution (synchronous vs asynchronous)
- Volume of data (may determine the type of integration used)
- Alerts and monitoring built in Logging of integration activity (for debugging/auditing purposes); log successes and errors and key information (i.e. volumes of data, timestamp, recipients, etc.)

Inbound

- Web services
  REST (preferred)
  SOAP

- APIs
  CURL
  Vendor open APIs
  REST
  SOAP
  XML-RPC
  JSON-RPC

ETL loads

File loads (i.e. upload of files to systems, move data from one system to another via files consumed by programs, FTP, SFTP, FTPs, SSH).
Specific shares are to be used, see DBAs when using this option.

Automated scheduling / scheduled tasks

Use of file shares and interface DB as required. DBAs are to be contacted for their input when this type of integration is used

Outbound
- Web services
  REST
  SOAP
- APIs
  CURL
  Vendor open APIs
  REST
  SOAP
  XML-RPC
  JSON-RPC

ETL loads

File loads (i.e. upload of files to systems, move data from one system to another via files consumed by programs, FTP, SFTP, FTPs, SSH).
Specific shares are to be used, see DBAs when using this option.
- Automated scheduling / scheduled tasks
  Use of file shares and interface DB as required. DBAs are to be contacted for their input when this type of integration is used

ANNEX 9: SYSTEM GOVERNANCE STANDARD

Purpose

The purpose of this document is to outline the systems development acceptable procedures which forms part of the Systems and Applications Standard.
Effective development processes are critical to the success of systems development projects. This Systems Development Governance Standard provides:

- Development standards for all stages of the System Development Lifecycle
- Minimum requirements for software development activities, deliverables and acceptance sign-off.

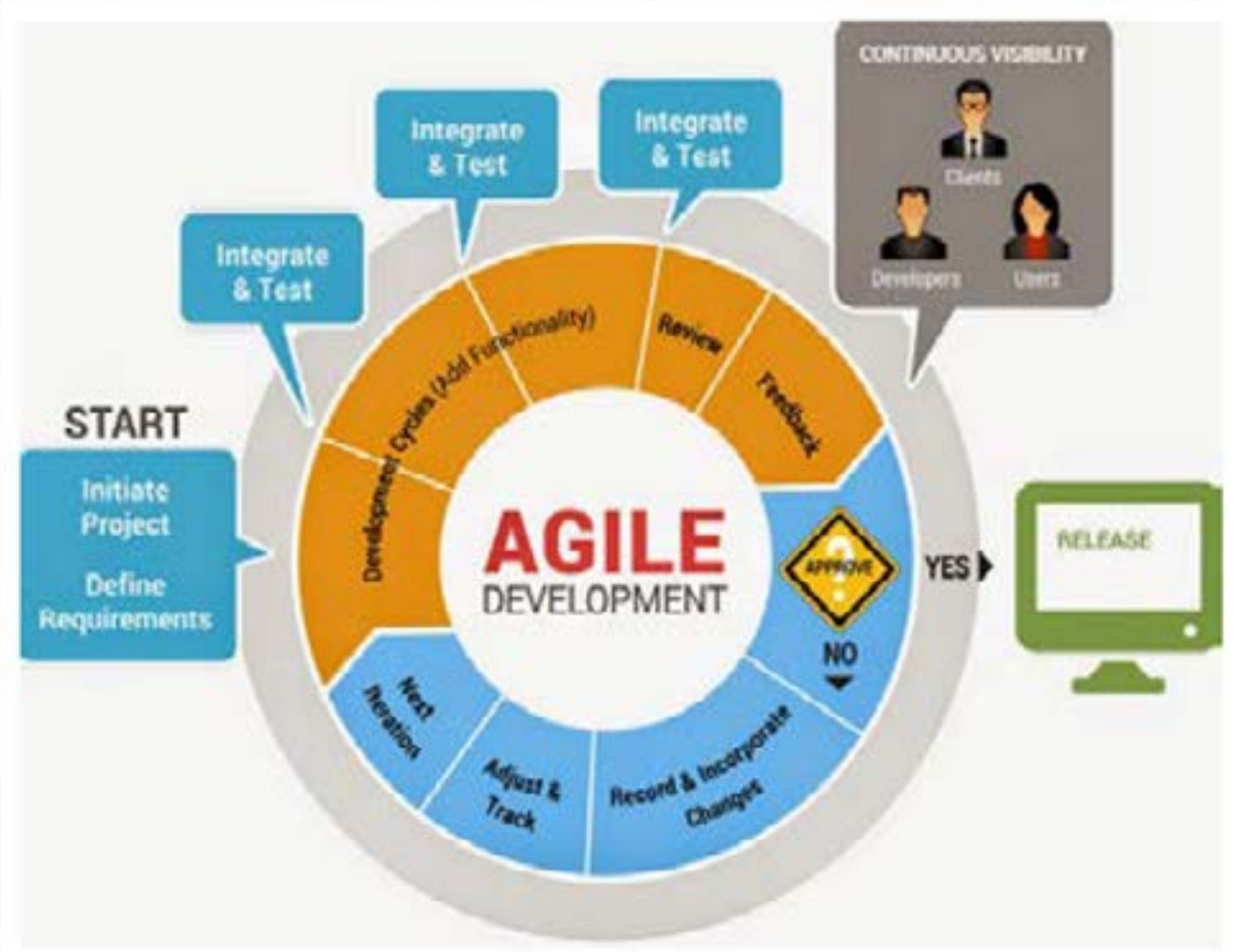
Scope

These requirements are mandatory and must be adhered to by all members of the development teams, consultants and / or contractors involved in the development or modification of mission critical applications that support the MCDA at an enterprise level.

Definitions, Acronyms and Abbreviations

Agile Method:

A software development method which breaks tasks into small increments with minimal planning and do not directly involve long-term planning. Iterations are short time frames that typically last from one to four weeks. Every iteration involves a cross-functional team working in all steps: planning, requirements/analysis, design, coding and testing. At the end of the iteration, a working product is demonstrated to stakeholders. This minimizes overall risk and allows the project to adapt to changes quickly. Iteration might not add enough functionality to be useable on its own, but the goal is to have an available release at the end of each iteration. Multiple iterations might be required to release a product or new features.



Application:

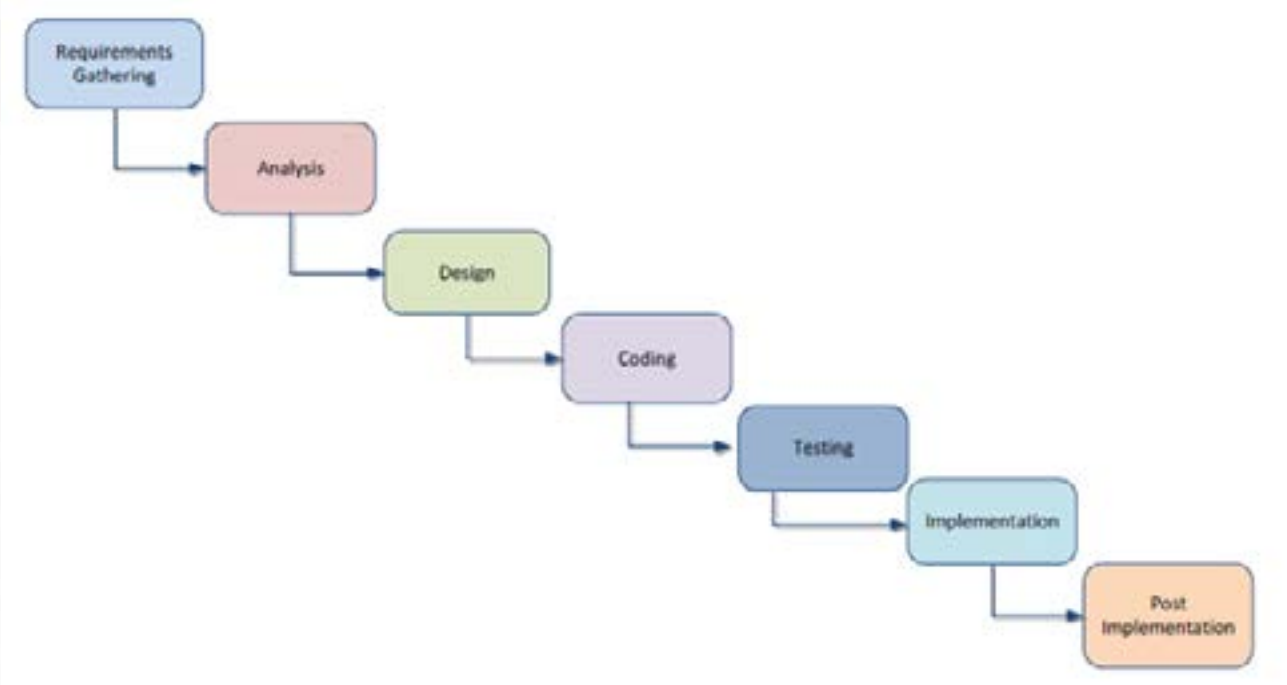
Computer programs, procedures, rules and associated documentation and data pertaining to the operation of a computer system.

Baseline:

A specification or end-product that has been formally reviewed and agreed upon. This becomes the basis for further development and must go through change control procedures to be altered.

Waterfall Method

A software development method. Waterfall is a sequential design process. Each phase in the lifecycle must be fully completed, documented and agreed upon prior to moving forward to the next phase.



System Analyst.

Cross-functional team: A group of people with different functional and technical expertise working together to achieve a common goal.
DBA: Database Administrator

Evaluation:

A technique in which requirements, design, code and test results are examined in detail by a person or group to detect potential problems. The results are documented.

Incremental / Iterative Methods:

A Software development method that breaks the project requirements into incremental changes (phases or sprints). The series of changes/releases is referred to as "increments", with each increment providing more functionality to the customers.
After the first increment, a core product is delivered, which can already be used by the customer. Based on customer feedback, a plan is developed for the next increments, and modifications are made accordingly.

Maintenance:

To repair, change or enhance software/application

Mission Critical:

A system or application whose failure will result in the catastrophic disruption of operations.

MoSCoW method:

A technique used to reach a common understanding with stakeholders on the importance they place on the delivery of each requirement. Also called the MoSCow prioritization or analysis.

Letter	Meaning	Description
M	Must	The requirement MUST be satisfied in the final solution for the final solution to be considered a success.
S	Should	A high priority item that should be included in the solution if possible. Often a critical requirement that may be satisfied in other ways if strictly necessary.
C	Could	The requirement is considered desirable but not necessary. It should be included in the solution if time and resources permit.
W	Won't	A requirement that stakeholders have agreed will not be implemented in a release but may be considered for the future.

Product:

A product is the tangible result of any process or work group. This includes (but is not limited to) purchased software products, plugins, components, code, services and deliverables.

Project Team:

A group of people (may include various departments) who collaborate and work together to deliver a software product.

Regression Testing:

The process of testing changes to software to ensure the changes have not adversely affected current system functionality.

Sign-off:

The declaration that the product has met expectations and been accepted by the governing body of the project as well as end users.

Software Development Lifecycle (SDLC):

A systematic approach to creating software applications. This cycle typically includes the following seven phases:

- Requirements gathering - collection of business requirements /needs
- Analysis - Business and requirements analysis
- Design - Architecture and application design
- Coding –Development/programming
- Testing – Quality assurance, bug fixes, error correction
- Implementation – Deploying (releasing) the application into the production environment for business use
- Post Implementation – maintenance and review.
- System:
- A set of programs which perform all functionality defined within a software application.
- Subsystem:
- A functionally related to a subset of the system.
- User Acceptance Testing (UAT):
- UAT testing is performed by the functional business groups that will be using the software in production. It is performed on a test/quality assurance system that is separate from the production environment.

Walkthrough:

A review process in which an individual(s) leads their peers through their work product. This is used to evaluate requirements, specifications, code, documentation, etc.

General Standards

All software development projects, including maintenance projects, must follow these standards.
Objectives for application development include:

- Clear definition of purpose
- Simplicity of use
- Ruggedness (difficult to misuse, should not crash when errors are encountered)
- Delivered on time and when needed
- Reliability
- Efficiency (fast enough for the purpose it was created)
- Minimum development cost
- Conform to standards
- Clear, accurate and precise user documentation
- Clear, accurate and precise technical documentation.

All production systems must have designated Owners and Custodians for the critical information they process in order to identify requirements and verify the final deliverables with signoff.

There must be a separation between the production, development and test environments. This will ensure that security is rigorously maintained for the production system, while the development and test environments can maximize productivity with fewer security restrictions. Where these distinctions have been established, development and test staff must not be permitted to have access to production systems.

All applications are reviewed at predetermined checkpoints of the SDLC by the Application Architect or their designate. Any deviations are reported, and corrective action is determined prior to the application being released to production.
Formal authorization indicating Standards have been met is required before a new or modified application can be released to production.
Throughout the entire project, special consideration must be given on an ongoing basis to capturing and implementing security and privacy requirements. The post-implementation review must reflect this.

Requirements Gathering

Document a clear statement of the current situation outlining problems and opportunities the software will address.

Review existing systems/processes.

Identify and document the issues with the current data/ system/process.

Document:

- New business needs
- New assumptions since initial project assessment
- Items that will not be solved with this software (outstanding issues)
- Integration points with other software
- Data storage requirements
- Data retention requirements

- Cross-functional support/input required to proceed with this project/request
- Legislative/contractual/security/privacy/access requirements
- Confidential data, access rights to this data and compliance requirements for this data (e.g., payment requirements must comply with relevant regulations; confidentiality rules for personal data, etc.)

- Roles required for security.
- Logging requirements (i.e., what needs to be captured in audit logs other than who updated, when updated, etc.).
- Reporting requirements.
- Training requirements.

Walkthrough and review of requirements:
- Walkthrough with end users
- Walkthrough with peer.

Obtain sign-off and approval from end user to ensure there is agreement on the requirements as documented.

Deliverables:
- Requirements document.

Analysis

Evaluate the documented requirements. During the evaluation process the following criteria must be considered and results of the evaluation documented:

- Requirement can be traced to business need
- Requirement is consistent with business need
- Testability of the requirements
- Can the requirement be implemented given the current architecture?

Establish and document software requirements:

- Functional and capability specification, including performance and design characteristics and environmental conditions under which the software is to perform
- Interfaces external to the software module/ component/ service
- Testing requirements
- Privacy and security specifications
- Data definition and database requirements
- User acceptance and implementation requirements for the delivered application, including:

  - Will data validation from loads/migrations be required?
  - Test results required for quality acceptance
  - Test results required for performance acceptance
  - How will ease of use/usability be measured?

Operation and execution requirements:

- How often will the application be accessed?
- Does this application rely on the existence of another module/ data?
- Storage needs and potential growth
- What tools or technologies will users need to access the system (e.g., VPN, special licensing, etc.)?
- Maintenance requirements.

Walkthroughs and reviews:

- Architecture checkpoint review
- Internal peer reviews.

Design
Document the following:

- Identify tasks, pages, reports, procedures and functions
- Identify common modules that will be used
- Define the control logic for each task and unit
- Identify database and access requirements
- Identify feeds into the system module (including those coming from other sources)
- Identify exports and outputs from this system/module
- Assess performance requirements
- Define backup / recovery requirements
- Define the "how's" for each process rule and edit item identified in the analysis. These will become the logic descriptions in detailed design
- Define exception handling
- Conduct session(s) with architects (application and data architects) to review and flesh out design requirements
- Conduct final design walkthrough
- Document testing considerations: these requirements include identifying data files accessed, tests and startup considerations
- Define how the user will access the system/modules

Deliverables:

- Design document
- Project work schedule
- Coding specifications to be given to developers

Transitional documentation for operations group.

Coding and Development

SQL and .Net are the standard development platforms for enterprise applications.
Stored procedures/functions are required for all Database access (server side). Client-side coding is kept to a minimum and only used if approved by Application Architect and Business Systems Analyst. This is a security measure used to prevent infusion attacks.

All code must be written with the following goals in mind:

- Maintainability; adaptable to cope with changing requirements. The following questions will help judge the maintainability code:

  - Can I find the code that is related to a specific problem or change?
  - Can I understand the code? Can I explain the rationale behind it to someone else?
  - Is it easy to change the code? Is it easy for me to determine what I need to change?
  - Can I make a change with only a low risk of breaking existing features?
  - If I do break something, is it quick and easy to detect and diagnose the problem?

- Dependability. Does the code meet its specification, i.e., "correct output for each possible input"
- Efficiency. Is the program efficient enough for the environment in which it is used?
- Usability.

New stored procedures must include all the elements in the SQL template. Templates are located centrally in the department share and accessible to all developers.
Functions must be authorized and created by a DBA. Functions can be resource intensive depending on use so require a DBA to ensure efficiency and that their use is warranted.
Triggers must be authorized and created by a DBA. Triggers can be resource intensive depending on use so require a DBA to ensure efficiency and that their use is warranted.
User controls must be created/maintained by a Development Team Lead(s).

Comments/tracking: each code module must contain:

- Name of the module
- Purpose of the module
- Brief description of the module
- Original author
- Original implementation date
- Change control list which identifies:
  - Date of change
  - Who authored the change
  - Footprint ticket or project number that initiated the request for change
  - Description of change
  - Sample execute statement
  - In-line comment of variables at declaration to identify purpose and use
  - In-line comment blocks to describe required functionality for code when logic is complicated or more involved than usual. This adds to maintainability, allowing others to quickly understand what is happening in the logic.

Create / update unit tests documentation (at minimum) for modules/ features.
Create / update help documentation associated with the change to the module/ feature.
Create/update technical documentation associated with the change to the module/ feature.
Naming conventions must be adhered to.
Code portability is a goal. Hard-coding is to be avoided whenever possible (it is recognized that in some cases this is unavoidable). Should hard-coding be unavoidable, it must be identified and reviewed with the Business Analyst for approval. The approval must be documented as part of the project documentation.

All development must complete a code walkthrough prior to being promoted to QA/Production. Walkthroughs should include: BSA and a Team Lead. DBAs should be included for complex SQL or when processing times need to be reviewed.

SQL & .NET coding standards

Templates exist for SQL procedures (insert, update, delete, select). These templates include code that must exist in all SQL procedures used for data retrieval/update. Application session information is required to identify who is retrieving/ updating/ deleting data. Procedures used to fill screen drop down lists do not require session information when the data retrieved is not sensitive.
For batch procedures special coding is used to identify a batch run.

Testing

Each software change requires testing. The degree of testing will vary depending on the size and complexity of the change.
Each test must be supported by documentation. For simple, low risk changes, this may take the form of screen shots captured and put into a Word document.
At minimum, there must be a document identifying what was tested and signed off by an analyst and the primary user indicating the change was successfully reviewed and tested prior to implementing to QA and production.
Consult/Include Operations staff in the test phase for transition training.
Document your test

Identify the type of test you are performing:

- Unit testing
- Regression testing
- Load testing
- User acceptance testing.

Identify the functionality/features being tested. Identify the functionality/ features NOT tested for clarity.
Identify who is required to participate in the testing their role in testing. Consider:

- Analyst
- Developer
- End user
- Other (e.g., DBA, Support team).

Identify what needs to be in place before testing can begin:

- Is there specific hardware that is required, or a specific configuration that needs to exist?
- Is there software required for this testing?
- Does data need to be loaded/set up?

Identify the Pass/Fail criteria for each item tested:

- Steps / schedule of activities: identify tasks and dependencies between the tasks.

Deliverables:

- Test logs: Show items tested with inputs and resulting outputs. This should be derived from the test cases that were identified and documented in the coding / development phase.
- Issues Log: Record issues encountered during testing and corrective action taken.

Summary:

Brief summary of results, metrics and any observations during testing, including participant input.

Approvals and sign-offs:

To validate the system/change has been tested thoroughly, sign-offs from the Business Systems Analyst, system owner and any other testers are required.
Document implementation plan

Clearly define communication plan:

- Communication plan to user community of new functionality
- Communication plan internally to IT staff required to support the system. Identify and communicate what constitutes post-implementation support vs. operational support to the project team and users of the system. This includes but is not limited to:
  Change Advisory Board (CAB) need to be notified of the impending implementation (when and what is impacted)
  Operations group for ongoing support of the product
  Infrastructure teams to ensure proper backup/recovery processes are in place when production is live
  Client services, so they are aware of new/altered functionality and potential impact to the user community.
- Clearly identify dates for implementation (i.e., if phases exist, each phase's implementation date).

Clearly identify tasks required to get the product implemented:

  Who is required to perform each task.
  The order in which tasks must be completed, including all dependencies
  Identify risks:
  Involved with the implementation
  If the implementation is not performed
  Contingency plans associate with the risks.
- Identify all data imports required
  Are imports to be scripted or manually applied and by whom.
- Identify system configuration required before the product can be used in a live environment. Consider:

Security roles
Secure document/file repositories
Scheduled backups.

Training:

What training is required for the users of the system?
Who will do the training?

Validation:

Who will validate system functionality once the product has been implemented?
Who will validate data accuracy once the product has been implemented?
Identify scripts or procedures to perform validations.

Define the length of the post-implementation phase:

4 weeks can be used as a rule of thumb on large projects (1 year in duration) or complex projects. Adjust accordingly for smaller projects. For shorter duration projects, 1 or 2 weeks should be sufficient The post-implementation phase is supposed to give the user time to use all features and functions while resources are available to support any unforeseen issues.

Go Live Decision

- Ensure the project team and primary user contact/primary stakeholders are in agreement with the implementation plan.
- Ensure the primary user contact/primary stakeholders agree with all that has been completed and tested.

Get agreement to move to production. Deliverables:

Implementation plan document
Transitional documentation for operations team.

Post- ImplementationThe post-implementation time is typically referred to as a stabilization period. Resources are still assigned to the project, typically with a lower percentage of time allowance, and available to make adjustments if issues arise.
Any work done at this point is to ensure the functionality and features that were in scope as outlined in the charter, requirements, specification and design documents are working as expected.

Changes to functionality and features are not part of this process, they would require a new project or enhancement requests be initiated through regular channels.
Maintain a log of issues
Identify the priority level of each issue (to be done in consultation with the system user and project team):

- Level 1: assign to team member(s) to resolve as part of the post-implementation of the project. This level can be further broken down to prioritize the sequence to work on if multiple issues exist with this priority level.
- Level 2: create support tickets to be handled as part of the Operations team's regular mandate.

Ensure Operations team is kept aware of the issues log.

Prepare a project closing report identifying lessons learned and milestones.
Hold project closure meeting and obtain signature on completion report to close project.

Deliverables:

- Issues and resolution log.
- Project closure report.

Templates

Templates for creating SQL procedures are to be used to create new routines. These templates are shells set up to include basic standards such as routine naming convention, comment block and application session coding.
Specifications / Requirements checklist

- Was thought given to the system administration functionality?
- Was thought given to error handling?
- Does the specification clearly divide the project into phases?
- Do all the phases have verifiable (and preferably undisputable) outcomes?
- Does the document refer to any related documents as specifically as possible? (Document title, revision, page number)?

If there are interfaces:

- Have the necessary data required for interfacing been identified?
- Is the maximum load (data and system usage) estimated?
- Are the security requirements specified?
- Are the operation and maintenance requirements specified (service transition)?
- Are the education/training requirements specified?
- Are the installation/migration requirements specified?
- Has there been a peer-to-peer review (walkthrough)?
- Has the application architect reviewed (walkthrough)?
- Have the requirements/specifications been agreed to and signed off by the user?
- Have reporting requirements been clearly identified?

Design checklist

- Is the design as simple as it can be?
- Are all the functions/features that are listed in the requirements covered?
- Are all assumptions, constraints, design decisions and dependencies documented?
- Have all reasonable alternative designs been considered, including not automating some processes in software?
- Does the design have features or functionality which were not specified by the requirements (e.g., are all parts of the design traceable back to requirements)?
- Does the design create reusable components if appropriate?
- Are modules well-defined including their functionality and interfaces to other modules?
- Interface details:
  Routine name, parameters and their types, return type, pre- and post-condition, usage protocol with respect to other routines

File name, format and permissions

- Are all major data structures described and justified?
- Are major data structures hidden with access functions/procedures?
- Is the database organization and content specified?
- Are all key algorithms described and justified?

- Are all major objects described and justified?
- Is the user interface modularized so that changes in it won't affect the rest of the program?
- Is a strategy for handling user input described, i.e., file input, manually entered through filters, etc.
- Are key aspects of the user interface defined?
- Are storage space use estimates and a strategy for space management described and justified?
- Is a strategy for handling I/O described and justified?
- Is a coherent error-handling strategy included?
- Are error messages managed as a set to present a clean user interface?
- Are necessary buy vs. build decisions included?
- Is this designed to accommodate changes/enhancements in future?
- Is any part over- or under-designed?
  Are the major system goals clearly stated?
  Does the complete architecture fit together conceptually?
  Is the top-level design independent of the machine and language that will be used to implement it?
- Are you, as a programmer who will implement the system, comfortable with the design?
- Design review and Walkthrough completed with architects (data and application)
- Instructions/documentation for transition to Operations team

Development / Coding checklist

- Does every input that comes from an untrusted source (i.e., typing into fields on a page, external systems) have associated error checking accounted for?
- Are all forms of validation done on the server side? (Only allow on the client side on an as needed basis)
- Stored procedures used as the method for data validation/delivery
- Is each coding module of sufficient size? Limit the size for readability and maintainability.

Use a 4 page rule of thumb. If the module is larger than 4 pages consider if it can be reduced in size or functionality can be split out; also consider the following:

Size and quantity of data that would need to be passed between routines
Number of temporary tables that would be needed
Any extra database reads/writes that would be required.

Does the code have the following?:

- Proper naming convention
- Purpose is documented
- Brief description
- Original author and date are identified
- Change control area showing date of change, reason, person who did it and associated project or ticket number
- Sample execute
- Unit test documented and repeatable
- Sufficient commenting exists throughout the code to make it readable and understandable (i.e., maintainable) and the comments match the actual code.

Testing

- Test case(s) have been identified
- Pass/Fail criteria has been identified for each test case
- Page/Report filters
- Data entered in filters is trimmed
- Each valid option is tested for each filter
- Invalid data entered to test error control
- Test security by changing roles.

APPENDICES

Appendix 1: Critical Systems in Government

No	SYSTEM	DESCRIPTION
1	IFMIS (Integrated Financial Management System)	Integrated Financial Management Information System(IFMIS) is an automated system that enhances efficiency in planning budgeting, procurement, expenditure management and reporting in the National and County Governments in Kenya
2	E-CITIZEN PLATFORM	This is a system for Kenyan Citizens and Foreign Residents to apply for Government to Citizen (G2C) services and pay via mobile money, debit Cards and e-Citizen agents.
3	IPPD	The integrated payroll and personnel management database (IPPD) is a system for managing Government employee records
4	IPRS	The Integrated Population Registration System (IPRS) is an initiative aimed at consolidating population registration information into a single database for ease of verification by both Government and private bodies.
5	I.D SYSTEM	The I.D system at the National Registration Bureau is used for secure production and issuance of secure identification documents, management of a comprehensive database of all registered persons and detection and prevention of illegal registration.
6	GDC Applications	The Government Data Centre (GDC) is designed for processing and storage of Government applications and data.
7	Applications on CCP	The County Connectivity project aims at ensuring that county government offices are connected to the internet and promote online services using telephones, emails and teleconferencing.
8	E-MAIL	This shared service is used for online communication for all Government officers

Appendix 2: SDLC ACTIVITIES AND OUTPUTS

	PHASE	ACTIVITIES	OUTPUT/DELIVERABLE
1.	PROJECT DEFINITION	1. Collection of information to determine if a project warrants the investment of personnel resources and funding.	1. Scope of the project prior to committing funding and resources, including the project timetable with milestone dates and resource estimates, and a formalized approval/authorization or disapproval of the project based on the project definition. 2. Identify the customer, user, mandate, and basic operating concept. 3. Identification of the program and project manager as well as projected costs for training and sustaining efforts after the project is completed. 4. Preliminary risk analysis and high level cost- benefit analysis to determine if the project has a favorable return on investment
2.	USER REQUIREMENTS DEFINITION	1. Review user requests 2. Meet with users to clarify requests 3. Modify request requirements 4. User approval of requirements	1. User requirements that clearly describe what part of the user process (activity) should be automated or enhanced, and the expected capabilities and features. 2. The key output of this phase is a summary document of user requirements that explains what the system is supposed to do 3. Explicit written User approval on requirements (template available) 4. Explicit written User approval or requirements

3	**HIGH LEVEL ANALYSIS AND DESIGN**	1. Research and documentation 2. Review high level analysis and design document with sponsors	1. The key output of this phase is a summary document of system/ data requirements that explains what the system should be built to, how data should be processed, and what technical or support requirements may exist. In addition, security and internal control related requirements are also developed as appropriate to the scope of the project. 2. Explicit written sponsor approval of agreed upon solution 3. Resources assigned to project by sponsor
4.	**DETAILED ANALYSIS AND DESIGN**	1. The analysis and design phase is a complex and critical step in determining which system design, based on systems engineering and technology analysis, meets the user and system requirements. 2. For nontechnical solutions, the design may simply be a support process to be implemented over time 3. The design may be presented as several options with trade-off analysis or a specific configuration, and may consist of Commercial-offthe-shelf (COTS) products (preferred approach) or customized development. 4. Procurement options and cost information should be identified as determined by resource requirements and the design.	1. The recommendation of what to do or buy in order to meet the user and system requirements. 2. Detailed analysis and design document approval 3. Preliminary implementation plan 4. Detailed specifications approval

5.	SYSTEM BUILD/ PROTOTYPE/BUILD	1. Create test scripts 2. Coding 3. Unit test 4. Verification, validation and testing 5. Code review 6. Update system test plan	1. Functioning Prototype 2. Updated system test plan 3. Preliminary implementation plan 4. Unit level test scripts with unit tests results documented
6.	TESTING	• System Testing • Integrated systems testing • User Acceptance • Pilot testing • Finalize implementation plan	• Completed system test • Test scripts with system testing results documented • Completed integrated systems test • Test scripts with integrated system testing results documented • Explicit written Sponsor and User Group approval of system test results (approval template available) • Test scripts with user acceptance testing results ' documented • Functioning application in pilot environment
7.	IMPLEMENTATION AND TRAINING	1. procure, receive, configure, and install the new or revised system 2. Training according to the training plan 3. Documentation 4. User acceptance 5. Stakeholder notification	1. Successful transition to the new system without service interruption
8.	POST IMPLEMENTATION	1. Periodic reviews 2. Maintenance and enhancements 3. Security evaluations 4. Setting KPI 5. Continuous improvement 6. Quality assurance and user satisfaction	1. Written sponsor approval 2. Closed request

GOVERNMENT SYSTEMS AND APPLICATIONS STANDARD WORKING GROUP
MARY KEREMA-ICTA
JUDY MULI- COB
SAMWEL ONYANGO- NEWKCC
SCHOLARSTICA CHEPKEMBOI-ICTA
KENNEDY CHONGWO- ICTA

ICT Authority
Telposta Towers, 12th Floor, Kenyatta Ave
P.O. Box 27150 - 00100 Nairobi, Kenya
t: + 254-020-2211960/62
Email: info@ict.go.ke or communications@ict.go.ke or standards@ict.go.ke
Visit: www.icta.go.ke
Become a fan: www.facebook.com/ICTAuthorityKE
Follow us on twitter: @ICTAuthorityKE


ICT
Authority