European Committee for Standardization
Comité Européen de Normalisation
Europäisches Komitee für Normung

CEN Workshop Agreement (CWA)
Rev. February 16, 2018

Best Practices for the Design and Development of Critical Information Systems

CEN Workshop Members

Initiators / Sponsors	PROLOGISM *(CHAIRMAN)*, CS COMMUNICATION & SYSTEMES, LA BANQUE POSTALE
Participants	ARMA International ASD EISIS Groupement des Cartes Bancaires INFOCERT NYSE EURONEXT Technology RexConseil THALES VOLANS Informatica
Secretariat	AFNOR

TABLE OF CONTENTS

1. WORKSHOP CONTEXT AND DOCUMENT PURPOSE 3
2. TERMS AND DEFINITIONS 4
2.1. DEFINITION OF A CRITICAL INFORMATION SYSTEM (CIS) 4
2.2. DEFINITION OF CIS REQUIREMENTS 5
Integrity *5*
Availability *5*
Performance *5*
Capacity *5*
Security *6*
Maintainability *6*
Resilience *6*
Usability *6*
2.3. ADDITIONAL COMMENTS AND SPECIFIC ISSUES 7
3. GLOBAL MODEL OF CIS REQUIREMENTS 8
3.1. OVERVIEW 8
3.2. BASIC ASSUMPTIONS 9
3.3. ECONOMIC DIMENSION 10
3.4. BENEFITS OF THE MODEL 11
3.5. INTERDEPENDENCIES OF REQUIREMENTS 12
4. FUNDAMENTAL PRINCIPLES FOR DESIGNING AND BUILDING A CIS 14
4.1. IDENTIFYING AND AGREEING UPON SERVICE PRIORITIES WITH STAKEHOLDERS 14
4.2. DEFINING SERVICE CONTINUITY REQUIREMENTS 15
4.3. IDENTIFYING AND AGREEING UPON WHAT SHOULD BE MONITORED 15
4.4. SETTING UP AN ITERATIVE PROCESS 16
4.5. ASSUMING THAT PROBLEMS WILL OCCUR DURING THE RUN PHASE 16
4.6. SETTING UP A CONTROL SYSTEM 17
4.7. PERFORMING RISK AND REQUIREMENT BASED TESTING 17
5. BEST PRACTICES FOR DESIGNING AND DEVELOPING A CIS 18
5.1. LIST OF BEST PRACTICES 18
5.2. MAPPING PRACTICES VS. PROJECT PHASES 19
5.3. MAPPING PRACTICES VS. CIS REQUIREMENTS 20
6. ANNEX 1 - BEST PRACTICES SHEETS 21
BPS # CIS-01 – MODULARITY 22
BPS # CIS-02 – FAILURE ANTICIPATION 24
BPS # CIS-03 – ERROR PROPAGATION PREVENTION 26
BPS # CIS-04 – BOTTLENECK IDENTIFICATION 28
BPS # CIS-05 – DEFENSIVE PROGRAMMING 30
BPS # CIS-06 – EXECUTION TIME LOGGING 32
BPS # CIS-07 – RESOURCE CONSUMPTION SURVEY 34
BPS # CIS-08 – EARLY CAPACITY PLANNING 36
BPS # CIS-09 – INDUSTRIALIZED TESTING 38
BPS # CIS-10 – FRIENDS AND FAMILY PROBES 40
BPS # CIS-11 – TRANSACTION ID 42
BPS # CIS-12 – ERROR CASE LOGGING 44
BPS # CIS-13 – DATA TIMESTAMPING 46
BPS # CIS-14 – SERVICE MONITORING 48
BPS # CIS-15 – SHARED LOG SERVICE 50
BPS # CIS-16 – RUNTIME REPORTING 52
BPS # CIS-17 – PKI-BASED TRACEABILITY 54
BPS # CIS-18 – EXTERNAL SECURITY AUDIT 56
BPS # CIS-19 – CRISIS MANAGEMENT 58
BPS # CIS-20 – RETENTION MANAGEMENT 60
BPS # CIS-21 – FAILURE MODE ANALYSIS 62
BPS # CIS-22 – COMPLIANCE WITH THE RELEVANT STANDARDS 64
7. ANNEX 2 - LIFE CYCLE PROCESSES 66
8. ANNEX 3 - REFERENCES 68
9. ANNEX 4 - WORKSHOP MEMBERS 69

1. WORKSHOP CONTEXT AND DOCUMENT PURPOSE

The purpose of this CEN/ISSS Workshop is to develop a first level European agreement on best practices for market players to ensure quality in designing, developing, maintaining and operating critical information systems, including both applications and infrastructure.

This CEN Workshop background, objectives, work program, workshop structure and resource requirements are defined in the Business Plan, version 2.0 dated March 6, 2007.

The resulting deliverable consists of the present CWA (CEN Workshop Agreement). This document provides guidelines for the design, development and maintenance of information systems requiring a high level of quality of service (including performance and availability).

The workshop addresses mission-critical Management (or Business) Information Systems. It does not cover mission-critical systems in the scientific, industrial (control-command, etc.) and embedded systems domains, for example. In those domains, practices and technologies already focus on "technical" requirements whereas their "functional" requirements are generally specific, stand-alone and dedicated to a limited set of specifications.

The lifecycle of an IT project can be divided into three phases: Design, Build, and Run. This CEN Workshop addresses the practices required in the Design and Build phases, with a particular focus on how those practices impact the Run phase.

Finally, the workshop addresses practices required to fulfil technical specifications (or quality of service requirements), i.e. "Build it right and make it efficient". It does not address the practices required for functional specifications (or business requirements), i.e. "Build the right thing".

2. TERMS AND DEFINITIONS

2.1. DEFINITION OF A CRITICAL INFORMATION SYSTEM (CIS)

A business information system aims at performing functions to achieve a desired result. There are critical business functions and non-critical business functions.
Once translated into a computer system, each critical business function will be performed by one or more critical applications and infrastructures.
In addition, one particular critical application may cover both critical and non-critical business functions.

For the purpose of clarity, we will use simple and common sense definitions in this document.

For the purpose of this document, a *critical information system* is defined as any application and/or infrastructure which performs one or more critical business functions, as well as a set of applications and infrastructure whose combination performs one or more critical business functions.

"A critical information system (CIS) is one whose quality of service is essential to the successful functioning of the organization in which it is used: a failure in its quality of service results in the failure of the entire information system and has significant impact on the operations of the organisation in which it is used."

For the IT specialist, this definition is interpreted as follows:
"Technical specifications for CIS requirements demand just as much if not more effort than those for the functional requirements."

A CIS must provide:

- A high level of quality of service during the Run phase.
- A high level of life cycle control during the initial and on-going Design, Build and Run phases.

Although risk management is an important issue as far as Critical Information Systems (CISs) are concerned, it is outside the scope of this CEN Workshop. We recommend referring to the following standards and set of best practices:

- ISO/IEC FCD 27005: Information technology - Security techniques - Information security risk management.
  Status: International standard under development (Not available presently).
- ISO/IEC 16085:2006: Systems and software engineering - Life cycle processes - Risk management.
  Status: Published International Standard.
- AS/NZS 4360: Risk management.
  Status: Australian / New Zealand published standard.

2.2. DEFINITION OF CIS REQUIREMENTS

This section provides, for the purpose of this document, specific definitions of requirements crucial to a CIS. For a given CIS, only one or just a few requirements may be a major concern.

Two types of specific requirements may be crucial for a CIS:

- Quality of service requirements (generally addressing the concerns of both business process owners, stakeholders and IT specialists):
  - Integrity
  - Availability
  - Performance
  - Capacity
  - Security[1]

- Quality of system requirements (generally addressing the concerns of IT specialists):
  - Maintainability
  - Resilience
  - Usability

2.2.1. *Integrity*

"The system shall render the service without errors and/or loss or corruption of data."

2.2.2. *Availability*

"An available system is one that renders the expected service at the desired time i.e. with the expected timeliness and priority."

Availability also covers the capability of a CIS to return to a normal condition within a specified time interval after a problem or a failure occurs.

Availability is different from continuity of service (IT survival plan, disaster recovery plan, business continuity plan, etc.).

2.2.3. *Performance*

"The system shall render the service in the best possible elapsed time, within the required limits, under specified circumstances"

2.2.4. *Capacity*

"The system shall render the service within the specified volume and flows limits, and it shall react appropriately in case of an overflow"

[1] *Security is a special requirement:*
- it is an essential requirement for any application, be it critical or not.
- the eight other requirements addressed in the CIS approach must be considered systematically when designing any security solution.

2.2.5. *Security*

It is recommended to refer to ISO 27001 (formerly 17799-2005): "Information security is the protection of information from a wide range of threats in order to ensure business continuity, minimize business risk, and maximize return on investments and business opportunities".

In the scope of this CWA, we will only address "a subset of security requirements which are interdependent with other CIS requirements identified here".

In addition, it is important to point out that CISs are particularly subject to attacks (from both inside and outside the organization).

2.2.6. *Maintainability*

"The ease with which the CIS maintenance (both corrective and evolutive) can be performed in accordance with prescribed quality of service requirements."

In the case of services requiring special responsiveness, introducing corrections or changes to the CIS shall be possible without disrupting service (e.g. hot deployment).

2.2.7. *Resilience*

"A CIS shall automatically resist unexpected events (incidents, delays, etc.) while controlling consequences."

A CIS shall be designed to provide features such as:

- Automatic failure mode.
- Task priority allocation.
- "Clean" shutdown of the application.
- Automatic restarts.
- Total control during the return to normal conditions.

The testability of borderline situations is a major concern when designing and building a CIS.

2.2.8. *Usability*

Providing a regular working system is not enough when it comes to a CIS. A CIS shall always be under control: "when running a CIS has to report on its state and to give evidence of its well being. It is mandatory to know how the CIS is working and what difficulties it encounters or has encountered".

It is preferable to check the CIS continuously in order to help ensure that it is well under control. This reporting information is valuable both:

- For building a real time dashboard in order to detect earlier any possible deviation,
- And "off-line" for calculating key quality of service indicators.

This allows for example to relate the CIS performance with its capacity, and to know how it reacts to overloading or any unforeseen event. It also covers requirements related to traceability (e.g. audit trails, etc.).

In addition, "a CIS can be easily managed without impacting the quality of service":

- Manipulations shall be risk-free (stopping/restarting the entire system and various components, etc.).
- On-site presence of highly skilled personnel shall be as reduced as possible.

2.3. ADDITIONAL COMMENTS AND SPECIFIC ISSUES

Besides the key requirements described above, there are additional features which could help clarify certain issues related to the CIS environment. These include:

- Longevity of data (records) and systems (what elements are put in place to assure the longevity of the data and the systems, and how to act when it becomes an issue): this is considered to be part of the "maintainability" requirement. Related best practices will benefit to the CWA.

- System flexibility (how well does the system respond to changing technical environments and changing user needs): this is considered to be part of the "maintainability" requirement. Related best practices will benefit the CWA.

- Transfer of existing data to a new system: this is not considered as an issue specifically related to CISs. It is possible that data transfer may be more complex for some CISs, for example those running 24/7 which implies hot deployment and migration.

- Role of staff handling applications (staff needs to have a good understanding of the risks and threats and to know how to handle a risk situation): this is considered to be part of the "usability" requirement. Related best practices will benefit the CWA.

- Integration of CISs with other business applications and information systems: this issue shall be addressed at the same level as the Design and Build of the CIS. Related best practices will benefit to the CWA.

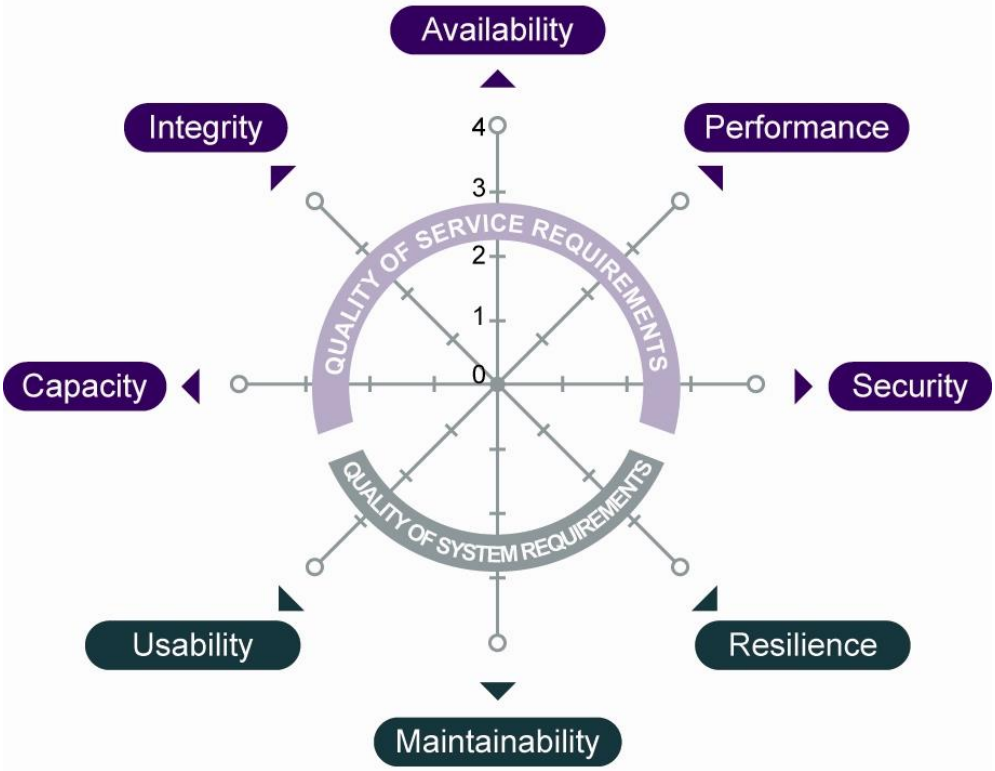
- Case of a system which relies on external service providers (e.g. telcos, outsourcers, etc.): the contract signed with the latter shall reflect CIS requirements. The contract shall allow the verification (audit) of the service provider in order to check that the requirements are met and the procedure implemented. This issue will not be addressed further in the scope of this CEN Workshop.

- So-called "external requirements" (how the CIS deals with external constraints such as European regulations, environmental and ethical standards, state-of-the-art in the industry, unwritten laws in its business, local country legislation, Sarbanes-Oxley Act, etc.): this issue is considered outside the scope of this CEN Workshop.

3. GLOBAL MODEL OF CIS REQUIREMENTS

3.1. OVERVIEW

The following CIS model (hereafter called the *CIS model*) formalizes the CIS requirements identified in chapter 2, as shown in the figure below.



This model may be used as a Kyviat diagram.

3.2. BASIC ASSUMPTIONS

"A combination of methods, techniques, best practices and tools contribute to satisfying the specific requirements of a CIS."

They shall be selected and applied according to the specific context of each project. Sections 4 and 5 describe certain techniques and best practices.

"The specific requirements of a CIS shall be specified by the stakeholder with the cooperation of IT people and taken into account as early as the specification and design phases of a system."

Experience shows that the later those requirements are taken into account, the more serious the consequences. The more "critical" the system, the more this rule applies.

These consequences can be divided into two broad categories:

- Those requiring a major re-design and overhaul of the system: uncontrolled increase in design and development costs, and delays in the rollout of the system.
- Those requiring the use of excessive human, material and software resources to operate the system in order to overcome its intrinsic weaknesses: uncontrolled increase in operating costs.

3.3. ECONOMIC DIMENSION

The CIS model also includes a crucial third dimension: the economic dimension (which does not appear in the figure shown in section 3.1 for simplicity's sake). The cost of the various solutions to be implemented shall be assessed according to the level of their achievements.

Besides initial design and development costs, companies shall consider the impact on costs for operations and future evolutions.

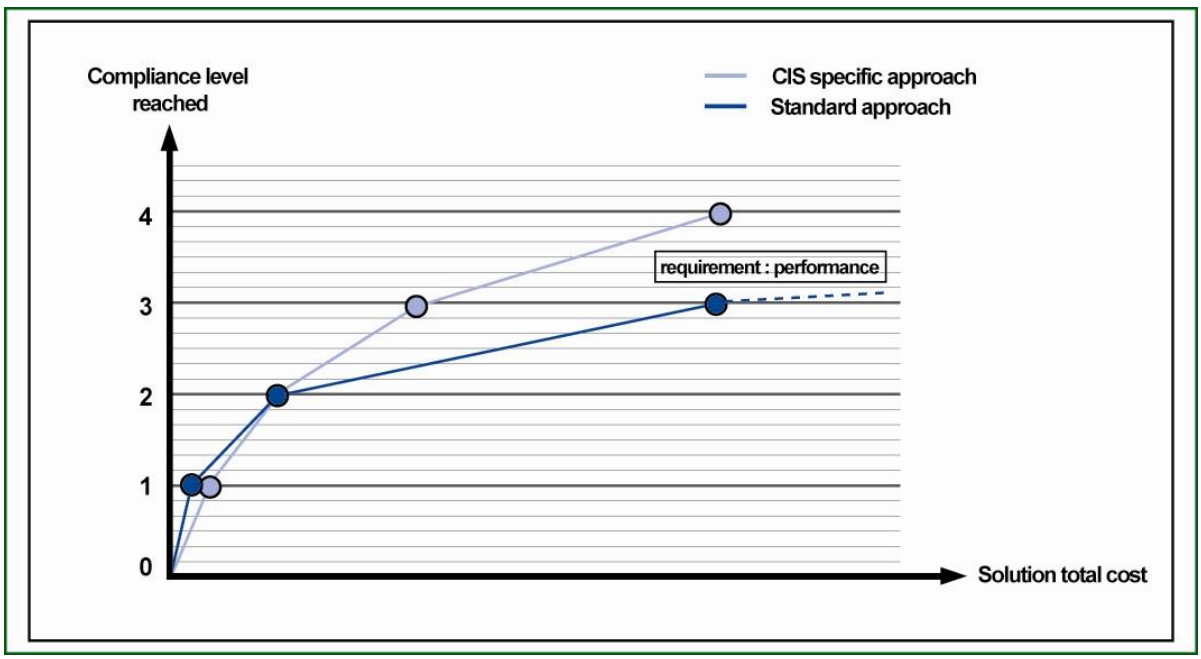
The approach to CISs based on the CIS model and the control of both the associated methods and the design and implementation techniques provide a global and "preventive" view making it possible to anticipate:

- The effects of a solution on all the criteria (see section 3.5 Interdependencies of requirements).
- Its impact in terms of costs (investment and recurrent costs).

Moreover, at equal cost, businesses will prefer a solution with a favourable impact on several of the CIS model's criteria.

Over the entire life cycle of a system, experience shows that this "systemic" method effectively addresses the CIS model's requirements and has lower total costs than a standard "analytical" approach.

Figure below illustrates this. The higher the compliance level, the more the company benefits from this approach to CISs.



3.4. BENEFITS OF THE MODEL

The CIS model measures to what extent a system complies with the CIS requirements, either:

When developing a new application:

- Representation of objectives and the target.
- Representation of relative priorities.

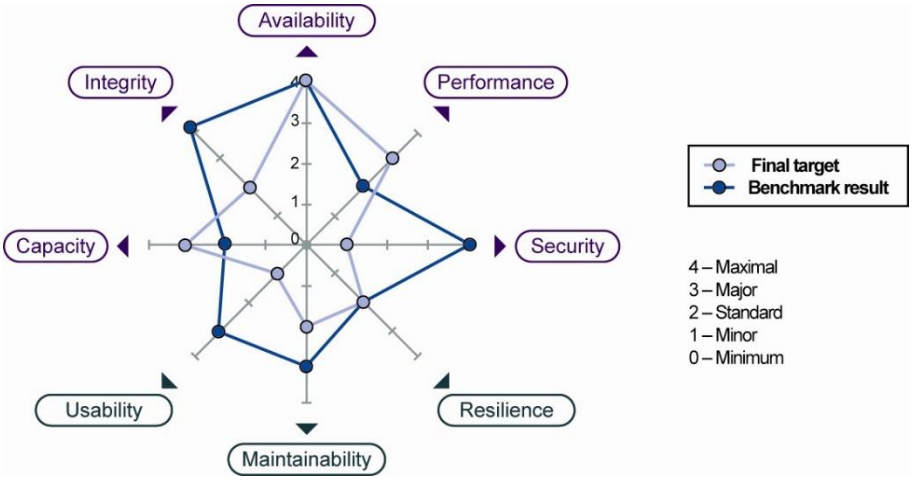
Or, when auditing an existing application, e.g.:

- Representation of how the system is perceived by various players (business sponsors, users, IT specialists).
- Representation of system audit results:
  - in the current operation situation,
  - in the projected ramp-up condition.

The figures below show two examples of reporting results:

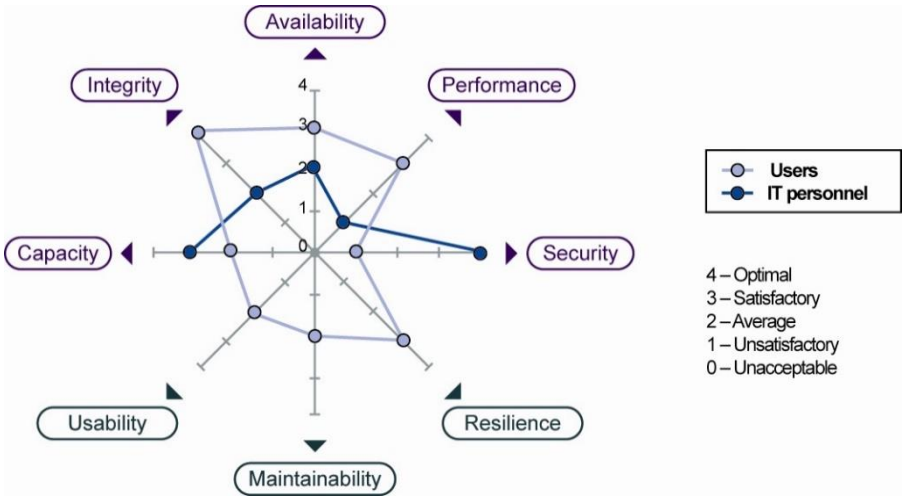
Development of the WebChain application: pre-integration phase

Intermediate results



Audit of the Focus application: Run phase

How various players perceive quality of service (synthesis)



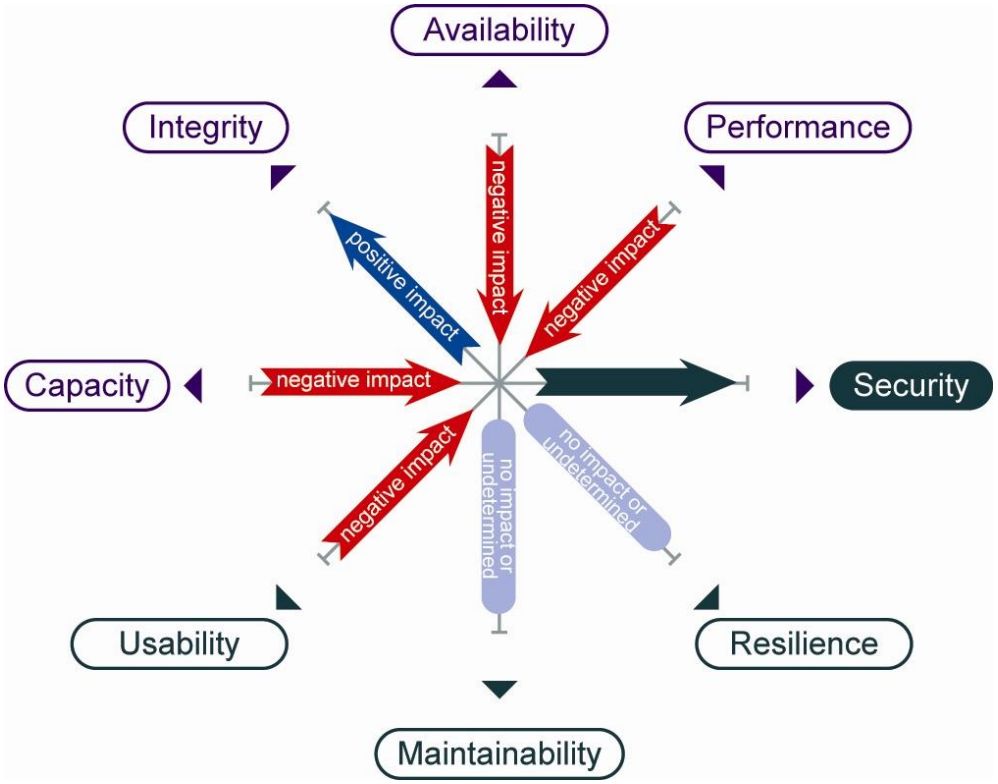
3.5. INTERDEPENDENCIES OF REQUIREMENTS

The methods and techniques to apply to ensure that a CIS complies with the above model's requirements are not mutually independent.

- For a given system, two requirements may be "antagonists": taking into account one requirement may, if one is not careful, result in an immediate or latent decline in the level of satisfaction with another requirement. The methodological approach is designed to counteract these effects.
- Conversely, for a given system, two requirements may be "convergent": taking into account one requirement may inadvertently result in an immediate or latent improvement in the level of satisfaction with another requirement. The methodological approach is designed to reinforce these effects.

There is no absolute rule of "systematic" antagonism or convergence between two requirements. Interdependencies between requirements vary for each project or system. However, experience shows that there are "natural" cause-and-effect relations between certain requirements. The figure and the table below illustrate the results of this experience.
This is a key consideration. It means that one cannot merely apply a method to suit each requirement, but one must adopt a "systemic" approach that addresses each requirement as much as its interdependencies with the other requirements.

Example: Impact of uncontrolled actions to improve one requirement over other requirements



Focusing exclusively on the security criteria can have:

- a positive impact on integrity
- a negative impact on performance, availability, capacity and usability
- a neutral or undetermined impact on maintainability and resilience

Another interesting example concerns performance. Although focusing on performance might have a negative impact on capacity (high volume of transactions and short response times are antagonistic *a priori*), capacity may nevertheless benefit from improved performance. Both assertions are arguable. One view is that performance improvements could cause capacity to be exceeded more quickly. On the other hand, they might also improve throughput capacity via faster processing.

The table below provides a succinct illustration of the "natural" antagonism and convergence between the specific requirements of CISs.
This table does not pretend to define a factual rule since there is no such absolute rules of "systematic" antagonism or convergence between two requirements. Its purpose is to help analyse these interdependencies and to structure the approach of those interdependencies in the scope of a project.
Note that the table is not symmetrical (the rows contain the criteria addressed, while the columns show their impact on other criteria).

When you focus on ↓ / What is the impact on →	Performance	Capacity	Availability	Integrity	Security	Usability	Maintainability	Resilience
Performance		0	0	0	0	0	0	0
Capacity	0		0	–	0	–	–	–
Availability	–	–		+	–	0	0	+
Integrity	––	––	+		+	+	–	++
Security	––	––	––	+		––	0	0
Usability	–	–	++	+	–		++	+
Maintainability	–	0	+	+	–	0		0
Resilience	–	–	++	+	+	++	0	

++	*often correlated in convergence*
+	*sometimes correlated in convergence*
0	*neutral position (no notable interdependency) or undetermined a priori*
–	*sometimes correlated in antagonism*
––	*often correlated in antagonism*

Identifying the interdependencies between requirements is a major factor in the economic evaluation of a project. For example, between two solutions with similar costs, one tends to choose the solution that simultaneously results in the immediate or latent improvement of several requirements ("sharing" of costs).

4. FUNDAMENTAL PRINCIPLES FOR DESIGNING AND BUILDING A CIS

As a general rule, satisfying all the requirements described above shall be the constant focus, and all the more so for CISs.

This focus concerns several major principles:

- Identifying and agreeing upon service priorities with stakeholders.
- Defining service continuity requirements.
- Identifying and agreeing upon what should be monitored.
- Setting up an iterative process.
- Assuming that problems will occur during the Run phase.
- Setting up a control system.
- Performing risk- and requirement-based testing (RRBT).

This list of principles is resulting from the workshop members thoughts, proposals and discussions. At the time of this writing, the seven principles outlined below were considered to represent a core set of principles for designing and developing a CIS. It does not pretend to offer an exhaustive set of principles.

4.1. IDENTIFYING AND AGREEING UPON SERVICE PRIORITIES WITH STAKEHOLDERS

The processes to ensure quality of services meeting the CIS requirements criteria are likely to differ according to the context of CIS projects, project development, project implementation, start of operations, current operations, and maintenance.

For stakeholders, the objective is to acquire and/or develop quality products (services) that satisfy user needs with measurable improvements to mission capability and operational support in a timely manner and at a fair and reasonable price.

Users' needs can be identified and expressed through the eight CIS requirements as essential elements of a CIS. Those can be achieved through the collaboration of skilled people and organizations representing stakeholders, with a clear mission and goal, with the right supporting information, with adequate human and technical resources, and producing the necessary documentation. Therefore, it is recommended to set achievable and clear priorities. The specifications can be achieved also through external activities such as call for tenders for example.

Whatever the context of the business case, an approval process shall be agreed between stakeholders and the IT department and formalized within a Service Level Agreement.

Note: RRBT method (refer to section 4.8) can also be used in the approval process to target the objective priorities.

4.2. DEFINING SERVICE CONTINUITY REQUIREMENTS

The service continuity requirements shall be defined at the same time as the other requirements. They will affect the system design and its cost.

It is essential for the Service/Business Manager and the IT Manager to reach a general agreement on acceptable system failures.

The accepted risks of failure will be balanced against the cost of the solutions to be implemented in order to ensure the desired service continuity:

- Maximum tolerable time window without service.
- Maximum time during which completed work may be lost and could have been redone.

As a result of this balancing approach, the solutions selected by the it manager may vary. For instance, it might be agreed that the system will not be explicitly designed to cope with multiple failure scenarios.

In this case, the designers will not need to study the possibility of two or three consecutive failures being able to cause the service to fail.

4.3. IDENTIFYING AND AGREEING UPON WHAT SHOULD BE MONITORED

In the case of a CIS, it is mandatory to be able to determine the status of the service and to have the capability of monitoring each component involved in providing that service.

Therefore, this requires to:

- Identify the critical indicators which determine how the service is performing.
- Determine what information would be required to monitor service performance.
- Consider trade-offs between the quantity and timeliness of the information and its impact on performance.

The monitoring of services shall not generate excessive statistics on components, making it impractical to analyse the results due to the sheer volume involved.

4.4. SETTING UP AN ITERATIVE PROCESS

In general, CISs are very complex, and it is difficult to evaluate ahead of time how the overall system will behave in limit conditions. For example, it is not unusual for test phases to reveal previously unknown limitations to components (middleware, OS, hardware, etc.).

This situation often results in the need to re-evaluate "under duress" certain choices made during the design or development phases, often late in the development cycle. Experience shows that in general the system must be adapted to these unexpected constraints, even if they are not attributable to the system.

As a result, one must try to anticipate iterations in the development process, in order to minimize the consequences. This implies paying particular attention to the system's modularity, in the broad sense of the term, during both the specification and design phases, as indicated in the table below.

Phase	Principles to be applied to ensure modularity
Specifications	Take technical constraints into account during the functional analysis of the system: ▪ Include an independent module for each potential bottleneck. ▪ Include an independent module for each common "technical service".
Design	Ensure the vertical separation of functions (separation of concerns). Ensure the horizontal separation of processing types. Render the modules as independent as possible (low coupling). Ensure high cohesion. Include multiple instantiation and physical delocation of each module.

4.5. ASSUMING THAT PROBLEMS WILL OCCUR DURING THE RUN PHASE

IT designers and developers should consider the following assertion as a postulate (i.e. as an "axiom"):

- The best expert in the world with the best technology in the world will never end up with a "zero default" system.
- Whatever the level of skill and quality involved in the Design and Build process, one must assume that failures are inevitable.

The Design & Build phases shall include the ability to anticipate these failures and to restart the system as quickly as possible when they occur.

The following issue needs to be addressed at the very beginning of a project: "How can we give the IT operations teams the ability to detect and anticipate quality of service problems which will occur inevitably, and when they do, to prepare them to deliver a quick diagnosis and react promptly to those problems?".

4.6. SETTING UP A CONTROL SYSTEM

Setting up a control system is highly recommended when dealing with CISs. Best practices such as these offer an operational alternative to reference models and standards, which are often too complex to implement.

A dedicated task during each stage identifies the requirements and means of setting up control systems. Those requirements can be divided into three categories:

- Knowledge of facts, events, faults, access controls, permissions to access data, etc., occurring in critical systems.
- Control of critical systems characteristics and behaviour (event log monitoring, alert, case-based reference model for diagnosis identification, etc.).
- Actions to provide means for corrective and/or preventive solutions.

Close collaboration of business and IT departments is a prerequisite for efficient problem-solving. The following actions shall be taken:

- Operation monitoring 24 hours a day / 7 days a week.
- Implementation of appropriate track and trace procedures.
- Appropriate crisis management procedures (i.e. anticipating backup solutions in case of complex operations such as systems migration, triggering alerts, etc.).

4.7. PERFORMING RISK AND REQUIREMENT BASED TESTING

In the case of CISs, the Risk and Requirement Based Testing (RRBT) method may be used as an enhancement of standard test methods.

Since CISs usually require a very high level of (often expensive) testing, the management of tests by requirement/risk allows Project Managers to share test coverage with the stakeholder and provide very good visibility on risks and costs.

RRBT consists of analyzing stakeholder requirements as early as possible in the development cycle – during the specification phase if possible. Requirements are sorted by the risk level attributed by the stakeholder and a value is assigned to each requirement. Choosing the best test coverage consists of balancing risks and costs to make the optimal choice.

RRBT is also appropriate for CIS maintenance, with the possibility of adding experience from past cases encountered during the Run phase.

Some restrictions apply, however:

- For best results, it is recommended to select this method before starting a new project.
- Requirements shall be well defined, otherwise it will be difficult to evaluate the risk and build a test plan.
- The RRBT method does not apply for testing in the scope of change management.

Note: The RRBT method is not issued from a national or international standards organization. However, when dealing with risk in the scope of RRBT, it is best to refer to ISO/IEC 16085:2006 (Systems and software engineering - Life cycle processes - Risk management).

5. BEST PRACTICES FOR DESIGNING AND DEVELOPING A CIS

5.1. LIST OF BEST PRACTICES

The following table presents a list of best practices which can be applied when designing and developing Critical Information Systems. This table outlines the best practices identified by the members of the Workshop which seemed to be most relevant to the Workshop goals and objectives.

It is important to note that, over time with the development of advanced and new technologies, new best practices will no doubt be developed. At the time of this writing, the 22 best practices outlined below were considered to represent a core set of best practices for designing and developing a CIS.

Each of the 22 practices is described in Annex 1 in a form called BPS (Best Practice Sheet).

BPS #	Title
CIS-01	Modularity
CIS-02	Failure anticipation
CIS-03	Error propagation prevention
CIS-04	Bottleneck identification
CIS-05	Defensive programming
CIS-06	Execution time logging
CIS-07	Resource consumption survey
CIS-08	Early capacity planning
CIS-09	Industrialized testing
CIS-10	Friends and family probes
CIS-11	Transaction ID
CIS-12	Error case logging
CIS-13	Data timestamping
CIS-14	Service monitoring
CIS-15	Shared log service
CIS-16	Runtime reporting
CIS-17	PKI-based traceability
CIS-18	External security audit
CIS-19	Crisis management
CIS-20	Retention management
CIS-21	Failure mode analysis
CIS-22	Compliance with the relevant standards

5.2. MAPPING PRACTICES VS. PROJECT PHASES

The following table shows which of the 22 best practices defined in this document are the most relevant for each phase in a project life cycle.

The various phases of a project life cycle listed here refer to the various processes and activities described in ISO/IEC 12207. However, for the purpose of the CWA, the list of activities as described by ISO/IEC 12207 is simplified. Some activities are considered as being of no interest here, while others are gathered in a broader activity, as shown in Annex 2.

Project life cycle phase		Best Practice Sheet concerned
Development process	Requirements analysis	CIS-01 CIS-02 CIS-03 CIS-04 CIS-08 CIS-14 CIS-16 CIS-17 CIS-19 CIS-20 CIS-22
	Architectural design	CIS-01 CIS-02 CIS-03 CIS-04 CIS-05 CIS-08 CIS-11 CIS-14 CIS-15 CIS-16 CIS-17 CIS-20
	Detailed design	CIS-01 CIS-02 CIS-03 CIS-04 CIS-06 CIS-11 CIS-13 CIS-14 CIS-15 CIS-16 CIS-17
	Coding and unitary testing	CIS-05 CIS-06 CIS-11 CIS-12 CIS-14 CIS-15 CIS-16
	Testing and integration	CIS-01 CIS-04 CIS-09 CIS-11 CIS-12 CIS-14 CIS-15 CIS-16 CIS-17
Operation process	Operational testing	CIS-03 CIS-04 CIS-06 CIS-08 CIS-09 CIS-10 CIS-11 CIS-12 CIS-14 CIS-15 CIS-16 CIS-17 CIS-21
	System operation	CIS-03 CIS-04 CIS-06 CIS-07 CIS-11 CIS-12 CIS-14 CIS-15 CIS-16 CIS-17 CIS-18 CIS-19 CIS-20 CIS-21
	User support	CIS-06 CIS-10 CIS-11 CIS-12 CIS-19
Maintenance process	Problem and modification analysis	CIS-03 CIS-04 CIS-05 CIS-06 CIS-07 CIS-11 CIS-12 CIS-14 CIS-15 CIS-19
	Modification implementation	CIS-01 CIS-02 CIS-03 CIS-04 CIS-05 CIS-06 CIS-12 CIS-15 CIS-16 CIS-17 CIS-18 CIS-22
	Maintenance review/acceptance	CIS-15 CIS-16 CIS-21 CIS-22
	Migration	CIS-04 CIS-09 CIS-10 CIS-17 CIS-18 CIS-20 CIS-21

5.3. MAPPING PRACTICES VS. CIS REQUIREMENTS

The following table shows which of the 22 best practices defined in this document are the most relevant when it comes to addressing each of the 8 CIS requirements identified by the Workshop.

CIS requirement	Best Practice Sheet concerned
Integrity	CIS-01 CIS-03 CIS-09 CIS-10 CIS-17 CIS-18 CIS-21 CIS-22
Availability	CIS-04 CIS-07 CIS-10 CIS-12 CIS-14 CIS-15 CIS-16 CIS-19 CIS-21 CIS-22
Performance	CIS-04 CIS-06 CIS-07 CIS-08 CIS-10 CIS-13 CIS-14 CIS-15 CIS-20 CIS-22
Capacity	CIS-04 CIS-06 CIS-07 CIS-08 CIS-13 CIS-14 CIS-22
Security	CIS-05 CIS-11 CIS-12 CIS-13 CIS-15 CIS-16 CIS-17 CIS-18 CIS-19 CIS-20 CIS-21 CIS-22
Usability	CIS-01 CIS-06 CIS-11 CIS-12 CIS-15 CIS-16 CIS-20 CIS-22
Maintainability	CIS-01 CIS-06 CIS-09 CIS-12 CIS-16 CIS-22
Resilience	CIS-02 CIS-03 CIS-04 CIS-05 CIS-09 CIS-15 CIS-18 CIS-21 CIS-22

6. ANNEX 1 - Best Practices Sheets

6.1. BPS # CIS-01 – MODULARITY

Best Practice Sheet # CIS-01 Modularity	
Practice summary	Increase maintainability by addressing modularity issues during the architectural design and implementation phases.
Purpose	Modularity criteria may be studied during the architectural design and implementation phases (including code and tests) to anticipate iterations in the development process and minimize their impact on functions and costs. Some techniques dealing with modularity are described below (non exhaustive list).
CIS requirement(s) concerned	☑ *Integrity* ☐ *Availability* ☐ *Performance* ☐ *Capacity* ☐ *Security* ☑ *Usability* ☑ *Maintainability* ☐ *Resilience*
Project phase(s) concerned	*Development Process:* ☑ *Requirements analysis* ☑ *Architectural design* ☑ *Detailed design* ☐ *Coding and unitary testing* ☑ *Testing and integration* *Operation Process:* ☐ *Operational testing* ☐ *System operation* ☐ *User support* *Maintenance Process:* ☐ *Problem and modification analysis* ☑ *Modification implementation* ☐ *Maintenance review/acceptance* ☐ *Migration*
Related deliverables	Methodological measures dealing with modularity included in the project set of standards (design method, programming rules, tests strategy)
Economic impact	*Initial costs:* ☐ *High* ☐ *Medium* ☑ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☑ *Low*
Risks if practice not applied	☐ *High* ☑ *Medium* ☐ *Low*

BPS # CIS-01 : Modularity *(continued)*	
Detailed description of the practice	Modularity involves maximizing independence and minimizing coupling between application functions or software components. The use of methods and tools facilitates a modular implementation. ■ During the architecture design phase: ▪ Use techniques for providing separation between independent software components, to contain or isolate functionalities and potentially reduce the effort of software verification process and maintenance process (i.e. partitioning techniques). ▪ Modularise application processing and optimise communication routes between modules to minimise coupling. ▪ Specialise functions (one module performs one function; a given function is only performed in one module) to avoid duplicating software components. ■ During implementation phases: ▪ Evaluate data coupling: a software component's dependence on data not exclusively under the component's control. The aim is to verify the components' order of execution and is based on their calling structure. ▪ Evaluate control coupling: the manner by which one software component influences the execution of another. The verifications concern the control of interfaces between the components to be integrated (module, sub-function, function) and the other components. ▪ Track dead code or unused variables during tests phases and remove it. Their removal precludes inadvertent execution of the dead code that may result in a system hazard. Dead code may be detected during structural coverage analysis (by using tests coverage tools). ▪ Detect duplicated code (by using static verification tools for example). These same principles of modularity and loose coupling apply to every level of the design: within a single code module, across an entire application, or across an entire system (for instance, a Service-Oriented Architecture embodies these principles).

6.2. BPS # CIS-02 – FAILURE ANTICIPATION

Best Practice Sheet # CIS-02 Failure anticipation	
Practice summary	Anticipate and plan for failures in Critical Information Systems.
Purpose	Complex systems shall expect failures and plan for them accordingly. This practice contributes to error containment.
CIS requirement(s) concerned	☐ *Integrity* ☐ *Availability* ☐ *Performance* ☐ *Capacity* ☐ *Security* ☐ *Usability* ☐ *Maintainability* ☑ Resilience
Project phase(s) concerned	*Development Process:* ☑ *Requirements analysis* ☑ *Architectural design* ☑ *Detailed design* ☐ *Coding and unitary testing* ☐ *Testing and integration* *Operation Process:* ☐ *Operational testing* ☐ *System operation* ☐ *User support* *Maintenance Process:* ☐ *Problem and modification analysis* ☑ *Modification implementation* ☐ *Maintenance review/acceptance* ☐ *Migration*
Related deliverables	
Economic impact	*Initial costs:* ☐ *High* ☑ *Medium* ☐ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☑ *Low*
Risks if practice not applied	☑ *High* ☐ *Medium* ☐ *Low*

BPS # CIS-02 : Failure Anticipation *(continued)*	
Detailed description of the practice	Complex systems shall expect failures and plan form them accordingly. This practice contributes to error containment. Planning for failures implies the system shall be designed so that losses are acceptable for users. The objectives are to: ▪ Minimize the damage caused by the failure => failure mode shall be retained within an isolated compartment ▪ Transmit information to system users during failure mode ▪ Minimize the time the system is in failure mode Planning for failures takes place during robustness (resilience) analysis included in the requirements analysis and architectural design phases. If requirement analysis is the "what" and architectural design is the "how," then robustness analysis is really preliminary design. This phase involves making some preliminary assumptions about the design, thinking about the technical architecture, and thinking through the various possible design strategies. So planning for failures is partly analysis and partly design. Some guidance for dealing with "planning for failure" is provided below (non exhaustive list): ■ Define requirements addressing anomalous behaviour by the system: ▪ Identify all the failure conditions of interfaced equipment, and in the absence of safety-related requirements at the system level, define derived "robustness" requirements. ▪ Perform this analysis on all input interfaces. ▪ Formally specify the detection mode, the signalling mode, and the appropriate sanction for each type of failure. ▪ Limit error handling to simple causes (no multiple causes). ▪ Identify the system's different functional modes (nominal modes, degraded modes*). ▪ Limit the number of degraded modes*: the system shall be easy to develop, maintainable, verifiable. ▪ Define performance for each degraded mode*. ▪ During the design phase, refine "robustness" requirements if necessary. ■ Study the architectural design type based on design constraints such as: ▪ Partitioning: technique for isolating one or more software compartments to prevent specific interactions and cross-coupling interference ▪ Redundancy ▪ Monitoring: functionality within a system which is designated to detect anomalous behaviour ■ At the end of design phase: Once robustness analysis is completed, carry out a review. This session helps to make sure that the robustness (resilience) requirements and the system architectural model are accurate and coherent. (*) *Degraded service* is defined as follows: Modules/applications/services ought to have a well-defined behaviour during partial or total failure of a dependent modules/applications/services.

6.3. BPS # CIS-03 – ERROR PROPAGATION PREVENTION

Best Practice Sheet # CIS-03 Error propagation prevention	
Practice summary	Avoid error propagation.
Purpose	Prepare an exhaustive inventory of all error cases and plan for the associated processing. Check incoming and outgoing flows (files, queries, etc.) in order to identify erroneous information and to control flow processing capacity.
CIS requirement(s) concerned	☑ *Integrity* ☐ *Availability* ☐ *Performance* ☐ *Capacity* ☐ *Security* ☐ *Usability* ☐ *Maintainability* ☑ *Resilience*
Project phase(s) concerned	*Development Process:* ☑ *Requirements analysis* ☑ *Architectural design* ☑ *Detailed design* ☐ *Coding and unitary testing* ☐ *Testing and integration* *Operation Process:* ☑ *Operational testing* ☑ *System operation* ☐ *User support* *Maintenance Process:* ☑ *Problem and modification analysis* ☑ *Modification implementation* ☐ *Maintenance review/acceptance* ☐ *Migration*
Related deliverables	
Economic impact	*Initial costs:* ☐ *High* ☑ *Medium* ☐ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☑ *Low* IT Team / Operations: facilitate incident handling by addressing the cause of the incident rather than its consequences.
Risks if practice not applied	☐ *High* ☑ *Medium* ☐ *Low* ▪ Significant increase in the number of incidents. ▪ Difficulties diagnosing the root of a problem. ▪ Services not rendered in the event of even a minor error.

BPS # CIS-03 : Error propagation prevention *(continued)*	
Detailed description of the practice	*Scope:* ▪ Architecture ▪ Design ▪ Load qualification ▪ Operation *Content:* ▪ List the application's incoming and outgoing flows. ▪ Prepare an exhaustive map of the application: detailed description of flows and flow processing centres. ▪ Make sure the functional specifications defining the flow content are complete. Each field shall be accurately defined (size, type, range of values, etc.). Define a default value in case of non-compliance. ▪ Make sure the mechanisms for switching to failure mode are identified if the incoming flow cannot be processed by the application or if the outgoing flow cannot be processed by the called application (e.g. in case it is blocked). ▪ Exhaustively define the processing required in case of an error that cannot be recovered automatically. *Means/resources:* Business Sponsor/IT team: Detailed functional specifications with exhaustive details on all the fields in all incoming and outgoing flows. Define processing in case of an error that cannot be recovered automatically. Architect: Define the technical certification scenarios: operation with a full production load, limit behaviour, break tests, failure mode. Development Team: Comply with the technical recommendations defined in the Detailed functional specifications (compliance with specified flows, programming of error handling). *Check:* During the technical certification phase: ▪ Simulate flows with errors. ▪ Check that error handling works correctly.

6.4. BPS # CIS-04 – BOTTLENECK IDENTIFICATION

Best Practice Sheet # CIS-04 Bottleneck identification	
Practice summary	Identify the bottlenecks in a system.
Purpose	Ensure smooth application flows by avoiding bottlenecks throughout the entire linkage chain.
CIS requirement(s) concerned	☐ *Integrity* ☑ *Availability* ☑ *Performance* ☑ *Capacity* ☐ *Security* ☐ *Usability* ☐ *Maintainability* ☑ *Resilience*
Project phase(s) concerned	*Development Process:* ☑ *Requirements analysis* ☑ *Architectural design* ☑ *Detailed design* ☐ *Coding and unitary testing* ☑ *Testing and integration* *Operation Process:* ☑ *Operational testing* ☑ *System operation* ☐ *User support* *Maintenance Process:* ☑ *Problem and modification analysis* ☑ *Modification implementation* ☐ *Maintenance review/acceptance* ☑ *Migration*
Related deliverables	
Economic impact	*Initial costs:* ☐ *High* ☐ *Medium* ☑ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☑ *Low*
Risks if practice not applied	☑ *High* ☐ *Medium* ☐ *Low* ▪ Random, incoherent increases in system capacities without analyzing the cause of the performance problem. ▪ If a critical component becomes saturated, restarting the service takes precedence over determining what caused the problem; steps taken to get the application back online may make it difficult to find the source of the malfunction. ▪ If a processor becomes saturated by a high workload, application response times suffer, and even service of certain application functions may be interrupted.

BPS # CIS-04 : Bottleneck identification *(continued)*	
Detailed description of the practice	*Scope:* ▪ Architecture ▪ Design ▪ Load qualification ▪ Operation *Content:* ▪ List the application's incoming and outgoing flows. ▪ Prepare an exhaustive map of the application: prepare a detailed description of flows and flow processing centres; divide flows into categories -- vital vs. non-vital, incoming vs. outgoing, long vs. fast processing. ▪ Define possible upstream bottlenecks, and any actions to be taken to diagnose drops in quality of service while the application is running. *Means/resources:* Business Sponsor: Statement of requirements to clearly identify the amount of information entering the system either synchronously or asynchronously. Business Sponsor / IT team: Specify the volumes and frequency of data entering and leaving the system. Technical Architect: Define the technical architecture, map the flows between modules, and define the system processing capacity. Set up a synchronous or asynchronous call system between modules for processing information. Define specific supervision indicators for data flows and for the processing capacity of the various processors. Be able to accept an extensive workload, mainly during the design phase (to detect all possible flows and to define the flow control mechanisms) and when qualifying system performance. Be present during the development phase to make sure the development team is correctly implementing the specifications. Development Team: Comply with the technical recommendations in the Architecture Document (comply with the specified flows, optimize processing time for complex calculations). Set up a buffer zone and a flow regulator. Perform load tests. Integration Team: Comply with the technical recommendations in the Architecture Document (configuration of buffer zones, compliance with the size of input and output connection pools). Optimize the number of system threads. Operator: Monitor the application and if necessary perform manual interventions to change flow regulator settings. Capacity Planning Manager: Increase the platform's processing capacity. *Check:* During the technical certification phase: ▪ Simulate a heavy incoming flow workload and check that the information is processed correctly at each node in the linkage chain (set up probes or technical counters). ▪ Check that the flow control regulators are working correctly.

6.5. BPS # CIS-05 – DEFENSIVE PROGRAMMING

Best Practice Sheet # CIS-05 Defensive programming	
Practice summary	Apply defensive programming practices
Purpose	Defensive programming rules are techniques that may be used to prevent code from unexpected execution. That kind of programming rules shall be defined before the coding phase within a coding standards document. Some examples are included below (non exhaustive list).
CIS requirement(s) concerned	☐ *Integrity* ☐ *Availability* ☐ *Performance* ☐ *Capacity* ☑ *Security* ☐ *Usability* ☐ *Maintainability* ☑ *Resilience*
Project phase(s) concerned	*Development Process:* ☐ *Requirements analysis* ☑ *Architectural design* ☐ *Detailed design* ☑ *Coding and unitary testing* ☐ *Testing and integration* *Operation Process:* ☐ *Operational testing* ☐ *System operation* ☐ *User support* *Maintenance Process:* ☑ *Problem and modification analysis* ☑ *Modification implementation* ☐ *Maintenance review/acceptance* ☐ *Migration*
Related deliverables	Programming rules
Economic impact	*Initial costs:* ☐ *High* ☐ *Medium* ☑ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☑ *Low*
Risks if practice not applied	☐ *High* ☑ *Medium* ☐ *Low*

BPS # CIS-05 : Defensive programming *(continued)*	
Detailed description of the practice	Some examples of defensive programming rules are included below (non exhaustive list). ■ Minimize input error data: ▪ Check errors or bounds in data input if it is from a human user or from another system ▪ Account for potential data buffer overflow conditions, and data aging ■ Avoid non-determinism: ▪ Minimize memory paging and swapping ▪ Limit the use of dynamic binding, memory allocation and deallocation ▪ Do not presume initial values for variables that are not explicitly initialised ■ Limit complexity: ▪ Minimize coupling ▪ Minimize the use of multi-tasking/multi-processing in the architecture ▪ Control the class library depth ■ Avoid interface errors: ▪ Minimize interface complexity (too many arguments, etc.) ▪ Avoid type coercion (implicit or automated type conversions, etc.) in interfaces between procedures ▪ Restrict global variable use ■ Avoid logical errors: ▪ Be sure to understand the possible loss of accuracy during computation ▪ Be mindful of conversion errors ▪ Avoid use of expressions with side effects ▪ Verify proper implementation of computed indices and array bounds ▪ Handle exceptions locally and uniformly

6.6. BPS # CIS-06 – EXECUTION TIME LOGGING

Best Practice Sheet # CIS-06 Execution time logging	
Practice summary	Log execution times. The modules shall offer transaction processing (I receive a question, I process it, I respond) that measures response times and logs the results.
Purpose	To be able to trace a transaction as closely as possible in order to improve performance. Analyze application behaviour under limit conditions (e.g. activity peaks) in order to identify points of contention
CIS requirement(s) concerned	☐ *Integrity* ☐ *Availability* ☑ *Performance* ☑ *Capacity* ☐ *Security* ☑ *Usability* ☑ *Maintainability* ☐ *Resilience*
Project phase(s) concerned	*Development Process:* ☐ *Requirements analysis* ☐ *Architectural design* ☑ *Detailed design* ☑ *Coding and unitary testing* ☐ *Testing and integration* *Operation Process:* ☑ *Operational testing* ☑ *System operation* ☑ *User support* *Maintenance Process:* ☑ *Problem and modification analysis* ☑ *Modification implementation* ☐ Maintenance review/acceptance ☐ *Migration*
Related deliverables	
Economic impact	*Initial costs:* ☐ *High* ☐ *Medium* ☑ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☑ *Low* Architecture: cost of defining development standards, and of defining the architecture. Developer: additional development work unless handled by a framework.
Risks if practice not applied	☐ *High* ☑ *Medium* ☐ *Low* Diagnosis is more difficult in case of a performance problem. Profiling tools need to be set up during operation. Poor application performance. Degradation of service.

BPS # CIS-06 : Execution time logging *(continued)*	
Detailed description of the practice	*Scope:* ▪ Design: Define the format of the log files, and execution times in particular ▪ Development: Trace significant milestones. ▪ Operation: Monitor response times, while taking care of avoiding overwhelming results and system overhead. *Content:* Application behaviour: ▪ The traces shall systematically include a field with a timestamp ▪ In debug mode, add more traces to fine tune the performance analysis *Means/resources:* ▪ The log system design document ▪ A test plan which contains the scenarios concerning response times for each transaction, on both a unitary and agglomerated/consolidated basis *Check:* ▪ Test results under load. ▪ Application behaviour during operation and diagnosis associated with response time problems

6.7. BPS # CIS-07 – RESOURCE CONSUMPTION SURVEY

Best Practice Sheet # CIS-07 Resource consumption survey	
Practice summary	Analyse the consumption of CIS resources. Monitor the consumption of the architecture's critical components under actual running conditions.
Purpose	Provide actual measurements of consumption in order to predict when a pre-determined critical value will be reached. Provide a monitoring sheet showing the state of resources for critical components. Set up a statistical approach in order to warn the IT Operations Manager of trends leading to a critical gap. Set up processes describing change management as soon a gap is reached.
CIS requirement(s) concerned	☐ *Integrity* ☑ *Availability* ☑ *Performance* ☑ *Capacity* ☐ *Security* ☐ *Usability* ☐ *Maintainability* ☐ *Resilience*
Project phase(s) concerned	*Development Process:* ☐ *Requirements analysis* ☐ *Architectural design* ☐ *Detailed design* ☐ *Coding and unitary testing* ☐ *Testing and integration analysis* *Operation Process:* ☐ *Operational testing* ☑ *System operation* ☐ *User support* *Maintenance Process:* ☑ *Problem and modification* ☐ *Modification implementation* ☐ *Maintenance review/acceptance* ☐ *Migration*
Related deliverables	Determination of critical gaps (input) Survey dashboard Statistical analysis survey Hardware Resource Survey Committee report (monthly, quarterly, etc.) IT department management process for hardware resources
Economic impact	*Initial costs:* ☐ *High* ☐ *Medium* ☑ *Low* *On-going costs:* ☐ *High* ☑ *Medium* ☐ *Low* The goal is to attempt to stay within the maximum yield zone, providing the ideal quantity of resources at the lowest possible cost, while fully complying with commitments. This yield shall be achieved with minimum risk for the project and specifically for the IT Operations Manager.
Risks if practice not applied	☐ *High* ☑ *Medium* ☐ *Low* Resources not available when required. Increased premium risk cost Poor cost management.

BPS # CIS-07 : Resource consumption survey *(continued)*	
Detailed description of the practice	During the capacity planning phase, determine the following with the IT Operations Manager: ▪ The gap triggering the process in question ▪ The processes for adjusting resources ▪ The analysis model for preparing the survey ▪ The dashboard survey ▪ A pre-defined roll-back in case of NOGO. This shall be carefully described for all players, particularly the Business Manager, the IT Operations Manager, and the operations team. The roll-back shall not impact the activity and it shall be secure. The entire process and its components shall be considered as standard parts of the project, and therefore shall follow the standard project life cycle.

6.8. BPS # CIS-08 – EARLY CAPACITY PLANNING

Best Practice Sheet # CIS-08 Early capacity planning	
Practice summary	Prepare a capacity plan during the feasibility phase.
Purpose	Provide realistic value for: ▪ Data ▪ Users access (average, hits/sec, etc.) ▪ Network flows ▪ Etc. in order to control: ▪ CPU usage estimate and scheduling ▪ Disk space estimate and scheduling ▪ User response times ▪ Network bandwidth estimate and scheduling ▪ Etc.
CIS requirement(s) concerned	☐ *Integrity* ☐ *Availability* ☑ *Performance* ☑ *Capacity* ☐ *Security* ☐ *Usability* ☐ *Maintainability* ☐ *Resilience*
Project phase(s) concerned	*Development Process:* ☑ *Requirements analysis* ☑ *Architectural design* ☐ *Detailed design* ☐ *Coding and unitary testing* ☐ *Testing and integration* *Operation Process:* ☑ *Operational testing* ☐ *System operation* ☐ *User support* *Maintenance Process:* ☐ *Problem and modification analysis* ☐ *Modification implementation* ☐ *Maintenance review/acceptance* ☐ *Migration*
Related deliverables	Capacity planning for each hardware item addressed
Economic impact	*Initial costs:* ☐ *High* ☑ *Medium* ☐ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☑ *Low*
Risks if practice not applied	☑ *High* ☐ *Medium* ☐ *Low* If resource shortages occur earlier than planned, urgent measures are required, which entail the following risks: ▪ Cost premiums ▪ Poor quality of service delivered and poor quality of maintenance ▪ User and client complaints Another risk is the oversizing of resources; this mainly has an economic impact.

BPS # CIS-08 : Early capacity planning *(continued)*	
Detailed description of the practice	A good estimation of hardware resources begins during the feasibility phase and is made jointly with the stakeholders (usually a Business department) who explain their requirements in terms of: ▪ Volume processed ▪ Grow or slowdown factors ▪ Users response time ▪ Availability period ▪ Etc. Then the capacity plan shall be adjusted to take those figures into account. This phase is conducted by the Project Manager and involves: ▪ The stakeholder ▪ The IT operator ▪ The Technical Architecture Department ▪ The Purchasing Department Capacity planning for each hardware item will provide guidelines throughout the project life cycle, enabling optimal resource scheduling (long-term procurement policy, correct equipment model, etc.) and providing the means to adjust to each part of the project if necessary. Capacity planning will be a major input of the performance tests phase, usually driven during the pre-production phase. Its early delivery can modify performance test scheduling by moving this stage forward. Capacity planning is also a good tool for mastering CIS scalability. It provides data for modelling the system's ability to cope with more activity, and points out potential limitations. Measurements can be used to build a resource consumption model in order to: ▪ Identify non linear components ▪ Identify limitations ▪ Automate capacity planning

6.9. BPS # CIS-09 – INDUSTRIALIZED TESTING

Best Practice Sheet # CIS-09 Industrialized testing	
Practice summary	Industrialize the testing process
Purpose	To guarantee that the coverage of tests is sufficient regarding the system. Tools that are needed: ▪ Test management system ▪ Functional testing tools ▪ Performance / capacity testing tools
CIS requirement(s) concerned	☑ *Integrity* ☐ *Availability* ☐ *Performance* ☐ *Capacity* ☐ *Security* ☐ *Usability* ☑ *Maintainability* ☑ *Resilience*
Project phase(s) concerned	*Development Process:* ☐ *Requirements analysis* ☐ *Architectural design* ☐ *Detailed design* ☐ *Coding and unitary testing* ☑Testing and integration *Operation Process:* ☑ *Operational testing* ☐ *System operation* ☐ *User support* *Maintenance Process:* ☐ *Problem and modification analysis* ☐ *Modification implementation* ☐ *Maintenance review/acceptance* ☑ *Migration*
Related deliverables	Tests plan and reports
Economic impact	*Initial costs:* ☐ *High* ☑ *Medium* ☐ *Low* *On-going costs:* ☐ *High* ☑ *Medium* ☐ *Low*
Risks if practice not applied	☐ *High* ☑ *Medium* ☐ *Low*

BPS # CIS-09 : Industrialized testing *(continued)*	
Detailed description of the practice	Buy and implement a Test Management system. Buy / develop functional testing tools. Buy / develop performance / capacity testing tools. Dedicate a team to testing from the beginning of the project. The testing team shall specify and implement the testing scenarios during the development phase. The tested system shall be as close as possible to the operational system. Create a tests library. Identify a subset as a core non-regression test suite. On each delivery, run the non regression test suite, plus the dedicated tests for the new features. Enrich the tests library with new tests whenever possible. Do not rely on people's memory. Remember to test the migration steps. Provide sufficient funding, people, and time for testing.

6.10. BPS # CIS-10 – FRIENDS AND FAMILY PROBES

Best Practice Sheet # CIS-10 Friends and family probes	
Practice summary	Check, in real run conditions, the application behaviour on a restricted and controlled panel of key users called "Friends and Family". For the maximum benefit, a specific organization is recommended, mainly based on Run phase tools and methods, to deal with any problems encountered as quickly as possible.
Purpose	For a CIS, perform the final control in real conditions. For the client, deliver a GO/NOGO instruction based on actual operating conditions. The increasing complexity of applications -- both in terms of interconnections and parameter specifications -- opens the door to new a category of risks: the ultimate Run phase adjustments. The purpose of this practice is to perform a final check before delivery.
CIS requirement(s) concerned	☑ *Integrity* ☑ *Availability* ☑ *Performance* ☐ *Capacity* ☐ *Security* ☐ *Usability* ☐ *Maintainability* ☐ *Resilience*
Project phase(s) concerned	*Development Process:* ☐ *Requirements analysis* ☐ *Architectural design* ☐ *Detailed design* ☐ *Coding and unitary testing* ☐ *Testing and integration* *Operation Process:* ☑ *Operational testing* ☐ *System operation* ☑ *User support* *Maintenance Process:* ☐ *Problem and modification analysis* ☐ *Modification implementation* ☐ *Maintenance review/acceptance* ☑ *Migration*
Related deliverables	Final validation before full service deployment
Economic impact	*Initial costs:* ☐ *High* ☐ *Medium* ☑ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☑ *Low*
Risks if practice not applied	☐ *High* ☑ *Medium* ☐ *Low*

BPS # CIS-10 : Friends and family probes *(continued)*	
Detailed description of the practice	Friends and family tests are complementary to pilot sites. Pilot sites are usually set up with specific resources, dedicated to this phase or using the target resources but restricted to a limited domain (few servers, no real data, etc.). The Friends and Family phase is different: it verifies the use of actual resources. The restrictions apply only to certain key users (the so-called friends and family). As early as the specification phase, schedule the Friends and Family test phase with both the Support Department and the stakeholders (end users, managers, etc.) in order to: ▪ Identify the main goal of the test – usually the stakeholder's final "Go/No go" ▪ Identify subsidiary goals for the IT department such as security checks, user access check, etc. ▪ Identify the method of checking (test): script, environment, testing conditions, list of restricted users, as well as their administration and duration. ▪ Identify acceptance criteria. ▪ Describe the treatment of incidents and errors encountered. Usually this kind of test involves: ▪ A short testing period. ▪ A period representing limited risk for the company (e.g. week-ends or evenings). ▪ A limited panel of well-prepared users. Often each user is given a specific test asset and a "real script", sometimes using their own client number and access. ▪ A specific communication and information organization. ▪ A short cycle for incident treatment, mainly based on the standard production method and tools.

6.11. BPS # CIS-11 – TRANSACTION ID

Best Practice Sheet # CIS-11 Transaction ID	
Practice summary	Include a transaction ID (business processing) in the trace files throughout the transaction life cycle.
Purpose	Enable a quick, detailed diagnosis of problems, respond to user support requests.
CIS requirement(s) concerned	☐ *Integrity* ☐ *Availability* ☐ *Performance* ☐ *Capacity* ☑ *Security* ☑ *Usability* ☐ *Maintainability* ☐ *Resilience*
Project phase(s) concerned	*Development Process:* ☐ *Requirements analysis* ☑ *Architectural design* ☑ *Detailed design* ☑ *Coding and unitary testing* ☑ *Testing and integration* *Operation Process:* ☑ *Operational testing* ☑ *System operation* ☑ *User support* *Maintenance Process:* ☑ *Problem and modification analysis* ☐ *Modification implementation* ☐ *Maintenance review/acceptance* ☐ *Migration*
Related deliverables	
Economic impact	*Initial costs:* ☐ *High* ☐ *Medium* ☑ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☑ *Low*
Risks if practice not applied	☐ *High* ☑ *Medium* ☐ *Low* Diagnosis is more difficult in case of an incident. Loss of time searching for information to help reproduce a case. Inefficient, unreliable user support may lead to distrust among users

BPS # CIS-11 : Transaction ID *(continued)*	
Detailed description of the practice	*Scope:* ▪ Design: In the log file format, include the "transaction ID" field ▪ Development: In each log message, update the transaction ID in the field designed for that purpose. ▪ Acceptance testing: Analyze the traces and be able to retrace the complete transaction using this ID. Test error cases and verify the traces *Content:* Enable to: ▪ Quickly isolate the desired transaction ▪ Track the transaction across the entire infrastructure ▪ Reconcile this transaction with all the application parameters *Means/resources:* ▪ Design document for the application monitoring system ▪ Test plan containing scenarios related to the presence of that ID ▪ Provide a monitoring system for acceptance test environments in order to verify that the diagnosis of failing cases is clearly established *Check:* ▪ Results of the technical acceptance testing. ▪ Application behaviour during operations and diagnosis associated with an incident For example: In the case of an Internet CIS (e-banking for example), one implementation is the following: the infrastructure delivers an ID to each "hit", and this ID is then carried all across the various components of the information system.

6.12. BPS # CIS-12 – ERROR CASE LOGGING

Best Practice Sheet # CIS-12 Error case logging	
Practice summary	Verify that the code includes a log message for each possible error case
Purpose	Quick diagnosis in case of an error
CIS requirement(s) concerned	☐ *Integrity* ☑ *Availability* ☐ *Performance* ☐ *Capacity* ☑ *Security* ☑ *Usability* ☑ *Maintainability* ☐ *Resilience*
Project phase(s) concerned	*Development Process:* ☐ *Requirements analysis* ☐ *Architectural design* ☐ *Detailed design* ☑ *Coding and unitary testing* ☑ *Testing and integration* *Operation Process:* ☑ *Operational testing* ☑ *System operation* ☑ *User support* *Maintenance Process:* ☑ *Problem and modification analysis* ☑ *Modification implementation* ☐ *Maintenance review/acceptance* ☐ *Migration*
Related deliverables	▪ Error message standardization document: format, content, numbering, severity, timestamp, etc. ▪ Best practices document for programmers with commented examples "good", "to be avoided". ▪ Acceptance testing scenario specific to identified error cases. ▪ Operating instructions with exhaustive coverage of all error cases ▪ Connectors to tools for third party alerts, analysis, or statistics
Economic impact	*Initial costs:* ☐ *High* ☐ *Medium* ☑ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☑ *Low* Costs generated: added cost under 5% during the initial phase, and subsequently trivial and quickly amortized at cruising speed. Expected gain after amortization. Stakeholders: architect, designer, integrator, operation
Risks if practice not applied	☐ *High* ☑ *Medium* ☐ *Low* Certain incidents go undetected, or are difficult to diagnose Inconsistent error management and multiplication of potential error cases

BPS # CIS-12 : Error case logging *(continued)*	
Detailed description of the practice	*Scope:* Development *Content:* Any function or method which might trigger an error shall return the error to the caller (in the form of a return or exception code). This function shall trace the error then either handle it, or return it to its caller. All detected errors shall include an explicit error message to facilitate diagnosis during development or operation. Error messages may be externalized or generated on the fly in the related code sections. Error messages produced by the system shall be collected (log files, tool, console, etc.). The error message format is standardized and compliant with common practice. For example, severity levels comply with Syslog standardization. There is a total of eight levels, identified by a number ranging from 0 (Emergency) to 7 (Debug): ▪ 0 Emergency ▪ 1 Alert ▪ 2 Critical ▪ 3 Error ▪ 4 Warning ▪ 5 Notice ▪ 6 Informational ▪ 7 Debug This indication is particularly important because it standardizes the representation of the severity of a log message, which for example enables interoperability between log collection and alert generation equipment. *Means/resources:* The acceptance testing phase shall not simply test passing cases. Each potential error case identified shall have a specific acceptance test sheet. Tools: ▪ Log collector, ▪ Syslog, ▪ Surveillance tools, ▪ Automatic analysis tools (e.g. Webalyzer, HDC Syslog). ▪ Automatic alert tool (e.g. sendmail) *Check:* Relevant tracing of all operation incidents. Stack trace in case of error. Verbose tracing in development mode for debugging

6.13. BPS # CIS-13 – DATA TIMESTAMPING

Best Practice Sheet # CIS-13 Data timestamping	
Practice summary	Perform data timestamping.
Purpose	To provide more meaningful and helpful information to clients and users by delivering key values together with their 'freshness' i.e. the time when they were entered into the system.
CIS requirement(s) concerned	☐ *Integrity* ☐ *Availability* ☑ *Performance* ☑ *Capacity* ☑ *Security* ☐ *Usability* ☐ *Maintainability* ☐ *Resilience*
Project phase(s) concerned	*Development Process:* ☐ *Requirements analysis* ☐ *Architectural design* ☑ *Detailed design* ☐ *Coding and unitary testing* ☐ *Testing and integration* *Operation Process:* ☐ *Operational testing* ☐ *System operation* ☐ *User support* *Maintenance Process:* ☐ *Problem and modification analysis* ☐ *Modification implementation* ☐ *Maintenance review/acceptance* ☐ *Migration*
Related deliverables	For significant fields, give date and time when the value was entered into the system
Economic impact	*Initial costs:* ☐ *High* ☐ *Medium* ☑ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☑ *Low*
Risks if practice not applied	☐ *High* ☐ *Medium* ☑ *Low*

BPS # CIS-13 : Data timestamping *(continued)*	
Detailed description of the practice	Example: At a bank, in the department producing account statements, set up a control report listing accounts that were not billed in due time due to incomplete information: ▪ for each account, give the date and time the account was last updated By the time the report is read, the missing information may have already been updated and completed. *_____ list of unbilled accounts on Friday, 19 October 2007 ____* *Account #: 012345678* *Owner's Name: John SMITH* *…* *Amount: €224.13* *Last update: 18/10/2007 22:54:57*

6.14. BPS # CIS-14 – SERVICE MONITORING

Best Practice Sheet # CIS-14 Service Monitoring	
Practice summary	▪ Formalize monitoring requirements ▪ Implement application monitoring
Purpose	▪ To know in real time how the CIS is performing under real conditions by collecting data on its state (how it is performing) ▪ Prepare Quality of Service management
CIS requirement(s) concerned	☐ *Integrity* ☑ *Availability* ☑ *Performance* ☑ *Capacity* ☐ *Security* ☐ *Usability* ☐ *Maintainability* ☐ *Resilience*
Project phase(s) concerned	*Development Process:* ☑ *Requirements analysis* ☑ *Architectural design* ☑ *Detailed design* ☑ *Coding and unitary testing* ☑ *Testing and integration* *Operation Process:* ☑ *Operational testing* ☑ *System operation* ☐ *User support* *Maintenance Process:* ☑ *Problem and modification analysis* ☐ *Modification implementation* ☐ *Maintenance review/acceptance* ☐ *Migration*
Related deliverables	Quality of Service indicators: performance and availability measurements
Economic impact	*Initial costs:* ☐ *High* ☑ *Medium* ☐ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☑ *Low*
Risks if practice not applied	☑ *High* ☐ *Medium* ☐ *Low*

BPS # CIS-14 : Service Monitoring *(continued)*

Detailed description of the practice

Monitoring an application consists in the real-time collection of:

- Key performance indicator (KPI) values
- Warnings and alerts

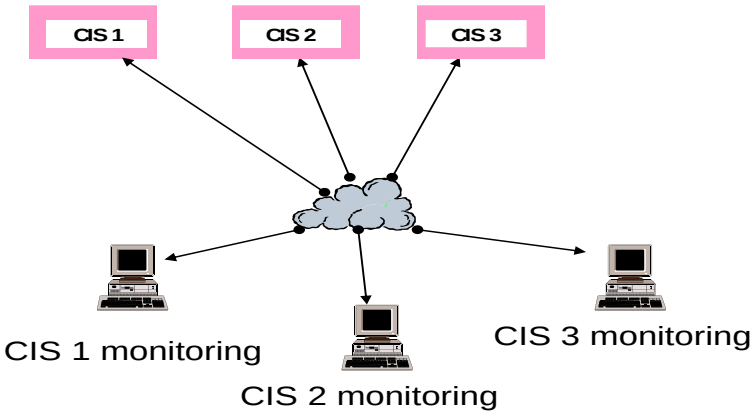
The reasons for monitoring a system are:

- to supervise the "health" (i.e. the correct performance) of the CIS in real time
- to supervise the client KPI in real time
- to help prevent failures and solve problems
- to reduce maintenance time for Design departments
- to reduce reporting time for IT Operations (Run) departments
- to ease system history management (keep track of KPI values, identify preventive action, etc.)

There are two ways to monitor a CIS:

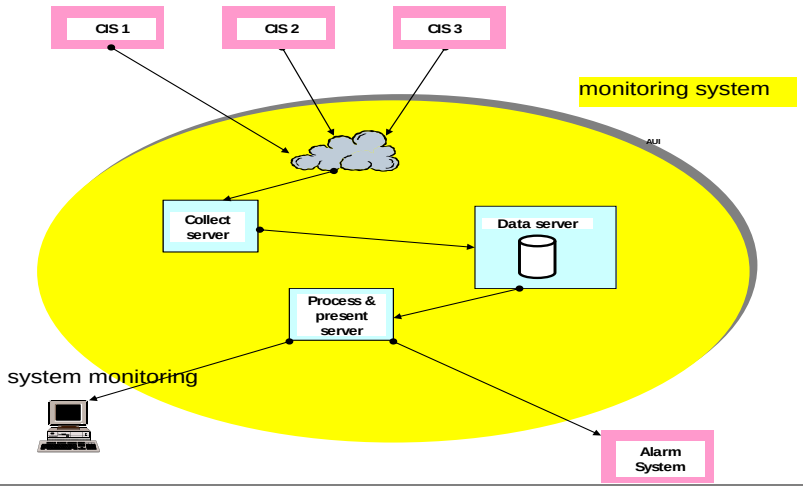
The simplest way is to use stand-alone computers to check the CIS, usually a PC running off-the-shelf testing software. Each computer regularly runs scripts as if it were an average user of the actual system.

Even if this is not the ideal solution for CISs, it has the advantage of helping to share views with stakeholders and IT operators.



A more mature solution is to require critical systems to continuously send information concerning their operations.

Naturally this means that the APIs with the monitoring system must have already been approved and implemented by the time the CIS is designed.



6.15. BPS # CIS-15 – SHARED LOG SERVICE

Best Practice Sheet # CIS-15 Shared log service	
Practice summary	Operational processes (related to a business or technical function) shall not trace information in a file
Purpose	Ensure that an independent architecture is set up to provide a shared log service for all processes.
CIS requirement(s) concerned	☐ *Integrity* ☑ *Availability* ☑ *Performance* ☐ *Capacity* ☑ *Security* ☑ *Usability* ☐ *Maintainability* ☑ *Resilience*
Project phase(s) concerned	*Development Process:* ☐ Requirements analysis ☑ *Architectural design* ☑ *Detailed design* ☑ *Coding and unitary testing* ☑ *Testing and integration* *Operation Process:* ☑ *Operational testing* ☑ *System operation* ☐ *User support* *Maintenance Process:* ☑ *Problem and modification analysis* ☑ *Modification implementation* ☑ *Maintenance review/acceptance* ☐ *Migration*
Related deliverables	
Economic impact	*Initial costs:* ☐ *High* ☑ *Medium* ☐ *Low* *On-going costs:* ☐ *High* ☑ *Medium* ☐ *Low*
Risks if practice not applied	☑ *High* ☐ *Medium* ☐ *Low* Diagnosis is much more difficult when an application crashes

BPS # CIS-15 : Shared log service *(continued)*	
Detailed description of the practice	*Scope:* ▪ Architecture: Implementation of a log collector responsible for their persistency, and of related tools (APIs) ▪ Development: Systematic use of this collector for tracing *Content:* Application behaviour: the trace mechanism is independent of the application. In other words, if the application crashes, the traces will still be available -- and are used precisely for this scenario. Set up a collector process and communication APIs with the collector. If the complexity of the existing architecture makes it expensive to set up a collector, then at least the trace files shall not be governed by a commit/rollback mechanism, so as to obtain the maximum amount of information in the trace files shall an application crash occur. *Means/resources:* Architect for defining the trace mechanism *Check:* Technical acceptance testing: ▪ robustness tests: crashing of processes, voluntary triggering of errors and verification in the log files

6.16. BPS # CIS-16 – RUNTIME REPORTING

Best Practice Sheet # CIS-16 Runtime reporting	
Practice summary	The application shall address incident diagnosis concerns (to repair as quickly as possible)
Purpose	▪ Detect technical incidents at runtime (resource problems, communication problems with external services, etc.) and provide the information operators need to restore the application service ▪ Provide a real-time diagnosis tool
CIS requirement(s) concerned	☐ *Integrity* ☑ *Availability* ☐ *Performance* ☐ *Capacity* ☑ *Security* ☑ *Usability* ☑ *Maintainability* ☐ *Resilience*
Project phase(s) concerned	*Development Process:* ☑ *Requirements analysis* ☑ *Architectural design* ☑ *Detailed design* ☑ *Coding and unitary testing* ☑ *Testing and integration* *Operation Process:* ☑ *Operational testing* ☑ *System operation* ☐ *User support* *Maintenance Process:* ☐ *Problem and modification analysis* ☑ *Modification implementation* ☑ *Maintenance review/acceptance* ☐ *Migration*
Related deliverable	
Economic impact	*Initial costs:* ☐ *High* ☐ *Medium* ☑ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☑ *Low*
Risks if practice not applied	☐ *High* ☑ *Medium* ☐ *Low* ▪ Diagnosis is more difficult, complex, and takes longer in case of incidents, ▪ Need to switch to debug mode during operation ▪ Long application unavailability periods ▪ No automatic incident recovery ▪ No accurate diagnosis for explaining incidents may lead to distrust in the system

BPS # CIS-16 : Runtime reporting *(continued)*	
Detailed description of the practice	*Scope:* ■ Design: Define an event collection and reporting mechanism suitable to the context: ▪ Standardization of traces ▪ Completeness of information ■ Development: At each event, update -- and verify the updating of -- trace files: add the error type, severity, and transaction ID in the appropriate fields. ■ Operation: Analyze the trace files and determine the cause of the failure P.S.: at this point we manage technical errors only, not application errors *Content:* The diagnosis of incidents shall follow two major themes: ■ Setting up of supervision indicators for both tracking activity and detecting any abnormal behaviour: drop in daily data/transaction volume, change in average response times, etc. ■ The trace details errors with in particular: ▪ standardized error logging (format, tools), ▪ explicit error messages about the type of error and the execution context. In the logging system, each transaction shall have a unique ID across all runtime modules, in order to enable detailed tracking. Thus it is necessary to include a field in the log files indicating the transaction ID. At every step in a transaction, traces shall systematically include a field indicating the transaction ID. *Means/resources:* Tools: ▪ Log collector (GES), ▪ Syslog, ▪ Surveillance tools, ▪ Automatic analysis tools (e.g. Webalyzer, HDC Syslog). ▪ Audit table ▪ Automatic alert tool (e.g. sendmail) Resources: ▪ Architect: design (rule definitions), validation of development ▪ Developer: Implementation of the rules ▪ Operator *Check:* Prepare a test plan including scenarios related to the presence of this ID in the traces Results of the technical acceptance testing: ■ Verification that each possible case of a technical error results in the requested trace The acceptance testing phase shall not simply test passing cases. Each potential error case identified shall have a specific acceptance test sheet. Application behaviour during operation and diagnosis associated with an incident

6.17. BPS # CIS-17 – PKI-BASED TRACEABILITY

Best Practice Sheet # CIS-17 PKI-based traceability	
Practice summary	Comply with traceability requirements related to Identity and Access Management (IAM) based on a Public Key Infrastructure (PKI).
Purpose	The objectives cover the following: 1) Identification and Authentication 2) Role Information assignment to the subject or object 3) Audit Logs
CIS requirement(s) concerned	☑ *Integrity* ☐ *Availability* ☐ *Performance* ☐ *Capacity* ☑ *Security* ☐ *Usability* ☐ *Maintainability* ☐ *Resilience*
Project phase(s) concerned	*Development Process:* ☑ *Requirements analysis* ☑ *Architectural design* ☑ *Detailed design* ☐ *Coding and unitary testing* ☑ *Testing and integration* *Operation Process:* ☑ *Operational testing* ☑ *System operation* ☐ *User support* *Maintenance Process:* ☐ *Problem and modification analysis* ☑ *Modification implementation* ☐ *Maintenance review/acceptance* ☑ *Migration*
Related deliverables	Security Policy Certificate Policy Certificate Practice Statement Operating Plan: backup and recovery procedures Documents relating to Target Of Evaluation (TOE) for each device or software supporting a qualified or certified process based on ISO/IEC 15408 with Protection Profiles or Security Target.
Economic impact	*Initial costs:* ☑ *High* ☐ *Medium* ☐ *Low* *On-going costs:* ☐ *High* ☑ *Medium* ☐ *Low*
Risks if practice not applied	☐ *High* ☑ *Medium* ☐ *Low*

BPS # CIS-17 : PKI-based traceability *(continued)*	
Detailed description of the practice	*Prerequisites:* ■ Refer to or develop a Security Policy ■ Refer to or develop a "Certificate Policy" and "Certificate Practice Statement" as a whole based on RFC 3647 *Context:* The general context is taken from the reference document "Certificate Issuing and Management Component (CIMC), Family of Protection Profiles that defines requirements for a PKI." *Definition:* A PKI is a security infrastructure that creates and manages public key certificates to facilitate the use of public key cryptography. To achieve this goal, a PKI shall perform two basic tasks: ■ generate and distribute public key certificates to bind public keys to other information after validating the accuracy of the binding; and ■ maintain and distribute certificate status information for unexpired certificates. CP: Certificate Policy: a CP is a document that describes the measures an organization will use to validate the identity of a certificate's subject. Validation might require a requestor-provided account and password combination submitted to the organization's directory or photo identification and submission to a background check through a registration authority (RA) process CPS: Certificate practice statement: a CPS is a public document that describes how a certification authority (CA) is managed by an organization to uphold its security and certificate policies. A CPS is published at a CA and describes the operation of the CA CRL: Certification Revocation List OCSP: Online Certificate Status Protocol *The PKI systems themselves shall be compliant with CIS best practices, in particular:* ■ Security: The main aspects of these tasks are relevant to the trustworthiness of the PKI and consequently rules and techniques ensuring strong Identity Management are essential to set up a secure information systems. For critical and sensitive applications all facets of information security shall be addressed, such as confidentiality, integrity, availability and non-repudiation. Organisations shall have Risk Management Plan in place if identity is compromised. ■ Availability: Resource availability is essential for operating Certificate Services and Identification & Authorisation modules in order to ensure business continuity. For example, to enhance the availability of Certificate Services, two or more issuing Certification Authorities shall share the enrolment process and provide verification services on certificate status (OCSP and CRL). This prevents Certificate Services from being unavailable due to a single point of failure. ■ Performance: Agreements shall be developed and implemented between end-users, application service providers, and credential issuers. These agreements will specify the various obligations and remedies for the participants in respects of liabilities, dispute solution, privacy and operational performance. *Focus:* ■ Identification and Authentication ■ Role Information ■ Audit Logs Ensure appropriate "track and trace" procedures implementation. Ensure archiving of identity and access management related data. Ensure compliance with CP, CPS or other applicable requirements (e.g. Common Criteria).

6.18. BPS # CIS-18 – EXTERNAL SECURITY AUDIT

Best Practice Sheet # CIS-18 External security audit	
Practice summary	Run security audits carried out by external experts.
Purpose	To guarantee sufficient coverage of system security requirements. Tools that are needed: ▪ hardware and software systems to prevent intrusion in the system ▪ security testing tools
CIS requirement(s) concerned	☑ *Integrity* ☐ *Availability* ☐ *Performance* ☐ *Capacity* ☑ *Security* ☐ *Usability* ☐ *Maintainability* ☑ *Resilience*
Project phase(s) concerned	*Development Process:* ☐ *Requirements analysis* ☐ *Architectural design* ☐ *Detailed design* ☐ *Coding and unitary testing* ☐ *Testing and integration* *Operation Process:* ☐ *Operational testing* ☑ *System operation* ☐ *User support* *Maintenance Process:* ☐ *Problem and modification analysis* ☑ *Modification implementation* ☐ *Maintenance review/acceptance* ☑ *Migration*
Related deliverables	
Economic impact	*Initial costs:* ☐ *High* ☐ *Medium* ☑ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☐ *Low* Initial costs and on-going costs are depending on the level of security desired (level of Common Criteria).
Risks if practice not applied	☑ *High* ☐ *Medium* ☐ *Low*

BPS # CIS-18 : External security audit *(continued)*	
Detailed description of the practice	A prerequisite to auditing the security requirements is to have defined those requirements: ■ Do a complete risk analysis of your system and its environment. ■ Define or describe the security policy relevant for your system including the security objectives for your system and its environment. ■ Describe the security requirements refined from the security objective (if possible use the Common Criteria at this stage to obtain a security target compliant with them). ■ Design the system and establish the mapping with the security requirements to argue how the threats are covered. ■ Test security functions when the system is developed (if possible perform a security evaluation following the Common Criteria). The goal of the audit is to verify that the organisational security procedures meet the organisational security requirements. In order to ensure proper coverage, it is essential that the external security experts/auditors be independent from the people who are building the system. The system being testing shall be very relevant compared to the operational system. Penetration tests shall be carried periodically (every 6 months or at least every year). The process described above shall be maintained to follow system evolution.

6.19. BPS # CIS-19 – CRISIS MANAGEMENT

Best Practice Sheet # CIS-19 Crisis management	
Practice summary	Set up a crisis management team. Design a crisis management procedure in case of major CIS breakdown.
Purpose	In the case of a CIS, it is important for stakeholders as well as for IT managers to decide -- as early as the Design phase -- on how to €cope with a major breakdown in order to react properly and safely. This best practice is not directly related to the IT process itself, but it reflects the general goal of dealing with CIS business control. This BPS is closely related to section 4.2 (Identifying and agreeing upon service priorities with stakeholders).
CIS requirement(s) concerned	☐ *Integrity* ☑ *Availability* ☐ *Performance* ☐ *Capacity* ☑ *Security* ☐ *Usability* ☐ *Maintainability* ☐ *Resilience*
Project phase(s) concerned	*Development Process:* ☑ *Requirements analysis* ☐ *Architectural design* ☐ *Detailed design* ☐ *Coding and unitary testing* ☐ *Testing and integration* *Operation Process:* ☐ *Operational testing* ☑ *System operation* ☑ *User support* *Maintenance Process:* ☑ *Problem and modification analysis* ☐ *Modification implementation* ☐ *Maintenance review/acceptance* ☐ *Migration*
Related deliverables	Crisis team process
Economic impact	*Initial costs:* ☐ *High* ☐ *Medium* ☑ *Low* *On-going costs:* ☐ *High* ☑ *Medium* ☐ *Low*
Risks if practice not applied	☑ *High* ☐ *Medium* ☐ *Low*

BPS # CIS-19 : Crisis management *(continued)*	
Detailed description of the practice	A specific procedure shall be prepared and the owner shall be clearly named for supervising a major breakdown. This supervision is generally allocated to the Run phase IT team which monitors the system. This team knows the rules that trigger the convening of the Crisis Team. The rules are mainly based on the user impact. Need for decisions is also an important criterion. In case of major problems, the crisis procedure is triggered. From that point, the Crisis Team makes all appropriate decisions and actions until the problem is completely solved. *Two kinds of teams are necessary:* ▪ Technical team ▪ Central command team *Goals of the teams:* ▪ Solve the breakdown or find a workaround ▪ Coordinate all the members, specifically concerning the run constraints ▪ Send a first-level alert to the predefined in-house players ▪ Assist members in the rerun phase, in order to correctly apply central command team decisions ▪ Report the breakdown ▪ Manage documentation related to the breakdown *Usually these teams do not:* ▪ Broadcast to outside players (media, etc.) ▪ Make decisions for technical or business stakeholders

6.20. BPS # CIS-20 – RETENTION MANAGEMENT

Best Practice Sheet # CIS-20 Retention management	
Practice summary	Implement system data and records retention policies and procedures
Purpose	To govern the appropriate retention of mission-critical data.
CIS requirement(s) concerned	☐ *Integrity* ☐ *Availability* ☑ *Performance* ☐ *Capacity* ☑ *Security* ☑ *Usability* ☐ *Maintainability* ☐ *Resilience*
Project phase(s) concerned	*Development Process:* ☑ *Requirements analysis* ☑ *Architectural design* ☐ *Detailed design* ☐ *Coding and unitary testing* ☐ *Testing and integration* *Operation Process:* ☐ *Operational testing* ☑ *System operation* ☐ *User support* *Maintenance Process:* ☐ *Problem and modification analysis* ☐ *Modification implementation* ☐ *Maintenance review/acceptance* ☑ *Migration*
Related deliverables	
Economic impact	*Initial costs:* ☐ *High* ☑ *Medium* ☐ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☑ *Low* The initial cost is low if an archiving infrastructure already exists. On-going costs include periodical reviews of system and procedures.
Risks if practice not applied	☑ *High* ☐ *Medium* ☐ *Low* If there are no retention rules, there is a risk that data will be deleted inappropriately. If information cannot be located and/or is incomplete, it could lead to unresolved audit queries and unsatisfied requests for information. This could undermine the trust that stakeholders have in the organisation, and ultimately have legal consequences (ENRON, WorldCom). On the other hand, if information of transitory value is kept unnecessarily, system performance might decrease and storage costs increase.

BPS # CIS-20 : Retention management *(continued)*	
Detailed description of the practice	Critical Information Systems contain information of both transitory and long-term value. It is essential to identify the records of great importance to the organisation not only to apply additional security and data integrity measures but also to guarantee long-term access to those records. On the other hand, keeping records of transitory value beyond their required retention time has a negative impact on system usability and performance. Therefore, it is essential that a CIS have rules based on the organisation's records retention policy to govern the timely deletion of transitory records and to keep an audit trail, while at the same time maintaining access to mission-critical records. The latter might imply upgrading to a newer version of the application, timely upgrades of technical hardware, and/or transfer to an archival repository. This includes the transfer of both the records and associated metadata in a standard file format. The key steps involved in the retention management are: ▪ In close consultation with the records or information managers, define the retention period for each record (data) group generated by the system. ▪ Link the retention period to the record (data) group and their associated metadata. ▪ When the retention period expires, apply the appropriate retention rule to the records groups, e.g., transfer of mission-critical records (data) to an archival repository when they are no longer needed for daily business, or permanently delete records (data) of short-term value when no longer needed for daily business and as evidence of actions and/or decisions. ▪ Keep an audit trail of the deletion and/or transfer of records (data). For more information on retention policies see ISO 15489, and on data preservation see ISO 14721.

6.21. BPS # CIS-21 – FAILURE MODE ANALYSIS

Best Practice Sheet # CIS-21 Failure mode analysis	
Practice summary	Perform a FMA (Failure Mode Analysis) for a new running system.
Purpose	To improve system resilience, degraded modes, and recovery times.
CIS requirement(s) concerned	☑ *Integrity* ☑ *Availability* ☐ *Performance* ☐ *Capacity* ☑ *Security* ☐ *Usability* ☐ *Maintainability* ☑ *Resilience*
Project phase(s) concerned	*Development Process:* ☐ *Requirements analysis* ☐ *Architectural design* ☐ *Detailed design* ☐ *Coding and unitary testing* ☐ *Testing and integration* *Operation Process:* ☑ *Operational testing* ☑ *System operation* ☐ *User support* *Maintenance Process:* ☐ *Problem and modification analysis* ☐ *Modification implementation* ☑ *Maintenance review/acceptance* ☑ *Migration*
Related deliverables	
Economic impact	*Initial costs:* ☐ *High* ☐ *Medium* ☑ *Low* *On-going costs:* ☐ *High* ☐ *Medium* ☑ *Low*
Risks if practice not applied	☐ *High* ☑ *Medium* ☐ *Low*

BPS # CIS-21 : Failure mode analysis *(continued)*	
Detailed description of the practice	It is good practice to perform a Failure Mode Analysis (FMA) on each service. This shall deliver an FMA report stating the recovery capabilities for all components, how they recover, and any known weakness in the recovery capabilities. This analysis shall take into account all the components of the service, including utilities (network elements, power supply, air conditioning, etc.). It might also contain recommendations as to how to improve service resilience of the service both from a system configuration/deployment viewpoint and from further systems development. The FMA shall be performed jointly by the IT Operations (Run phase) Department and the Technical Architecture Department: ▪ At the implementation of a new service (or after a major release) ▪ When actual system availability does not meet the requirements Refer to IEC 60812 for more details about FMA.

6.22. BPS # CIS-22 – COMPLIANCE WITH THE RELEVANT STANDARDS

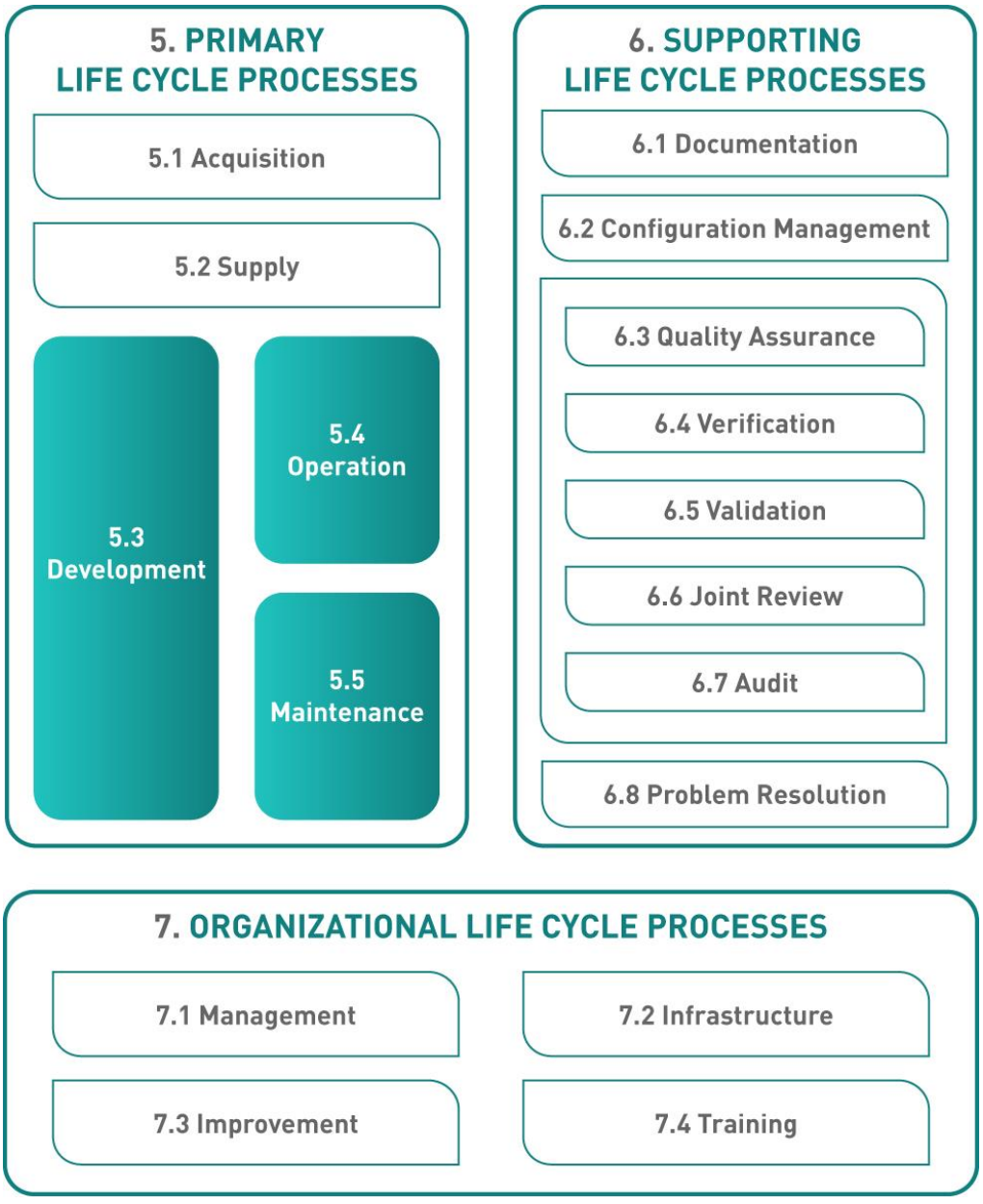
Best Practice Sheet # CIS-22 Compliance with the relevant standards	
Practice summary	The more critical the information system, the more important it is to comply with the relevant set of standards and best practices.
Purpose	Based on a set of standards to be applied, evaluations shall be carried out at every stage of the development process to verify that all guidelines are taken into account. The objective is to detect defects at the "right stage" and limit rework costs.
CIS requirement(s) concerned	☑ *Integrity* ☑ *Availability* ☑ *Performance* ☑ *Capacity* ☑ *Security* ☑ *Usability* ☑ *Maintainability* ☑ *Resilience*
Project phase(s) concerned	*Development Process:* ☑ *Requirements analysis* ☐ *Architectural design* ☐ *Detailed design* ☐ *Coding and unitary testing* ☐ *Testing and integration* *Operation Process:* ☐ *Operational testing* ☐ *System operation* ☐ *User support* *Maintenance Process:* ☐ *Problem and modification analysis* ☑ *Modification implementation* ☑ *Maintenance review/acceptance* ☐ *Migration*
Related deliverables	Set of standards and recommendations Evaluations results (reviews, audits, etc.) Corrective actions
Economic impact	*Initial costs:* ☐ *High* ☑ *Medium* ☐ *Low* *On-going costs:* ☐ *High* ☑ *Medium* ☐ *Low*
Risks if practice not applied	☐ *High* ☑ *Medium* ☐ *Low*

BPS # CIS-22 : Compliance with the relevant standards *(continued)*	

Detailed description of the practice	1. Define a set of standards for the project to be applied during the development process. This set of standards shall be establish by tailoring an organisational asset and shall be consistent with project and customers constraints. A minima it contains:
	- A Management Plan, including organisational and management dispositions, and methods, standards and tools for the development process, the configuration management process, and the assurance quality process.
	- A Verification Plan, including methods and tools for tests, reviews and analysis. Especially, it defines:
	  - Formal criteria to be used during evaluations: what will be evaluated, when or how often, how the evaluation will be conducted, who shall be involved.
	  - The test strategy, including testing methods based on requirements, requirements coverage analysis average (usually 100%) and/or structural coverage analysis (depending on the security requirements).
	
	2. Apply the project set of standards during the entire development process, including the drafting of a set of technical documents. Produce the traceability matrixes, including functional, technical, and methodological requirements, and verify they are compliant and complete.
	
	3. Perform objective evaluations based on formal criteria:
	- Formal audits by organisationally separate quality teams
	- In-depth review of work at the place it is performed (i.e. desk-audits)
	
	4. Implement peer reviews process as an objective evaluation method:
	- Members are trained and roles are assigned
	- Verification checklists are available
	- Peer reviews may be performed at each stage of the project
	- Defects are recorded, tracked, and escalated outside the project (i.e. customer) when necessary.

7. ANNEX 2 - Life Cycle Processes

When this document discusses life cycle processes, it refers to the various processes and activities described in ISO/IEC 12207. The domains of interest for the purpose of the CWA are filled in green on the following chart which is an excerpt from ISO/IEC 12207.



For the purpose of the CWA, the list of activities as described by ISO/IEC 12207 is simplified. Some activities are considered as being of no interest here, while others are included in a broader activity, as shown in the table below.

Development processes	
Taxonomy of activities as described by ISO/IEC 12207	**Taxonomy of activities simplified for the purposes of this CWA**
1. Process implementation	*Outside scope*
2. System requirements analysis	Requirements analysis
3. System architectural design	
4. Software requirements analysis	Architectural design
5. Software architectural design	
6. Software detailed design	Detailed design
7. Software coding and testing	Coding and unitary testing
8. Software integration	Testing and integration
9. Software qualification testing	
10. System integration	
11. System qualification testing	
12. Software installation	*Outside scope*
13. Software acceptance support	*Outside scope*

Operation processes	
Taxonomy of activities as described by ISO/IEC 12207	**Taxonomy of activities simplified for the purposes of this CWA**
1. Process implementation	*Outside scope*
2. Operational testing	Operational testing
3. System operation	System operation
4. User support	User support

Maintenance processes	
Taxonomy of activities as described by ISO/IEC 12207	**Taxonomy of activities simplified for the purposes of this CWA**
1. Process implementation	*Outside scope*
2. Problem and modification analysis	Problem and modification analysis
3. Modification implementation	Modification implementation
4. Maintenance review/acceptance	Maintenance review/acceptance
5. Migration	Migration
6. Software retirement	*Outside scope*

8. ANNEX 3 - References

- Christina BOLCHINI and others: The design of reliable devices for mission-critical applications - IEEE Transactions on Instrumentation and Measurement, *Volume 52, Issue 6, Dec. 2003, pages 1703-1712*
- Mario R. BARBACCI: Taxonomy - Software Engineering Institute - Carnegy Mellon
- ISO 27001: Information Security Management System
- ISO/IEC 12207: Information Technology - Software Life Cycle Processes
- ISO/IEC FCD 27005: Information technology - Security techniques - Information security risk management - State: International standard under development (presently not available)
- ISO/IEC 16085:2006: Systems and software engineering - Life cycle processes - Risk management - State: Published International Standard
- AS/NZS 4360: Risk management - State: Australian / New Zealand published standard
- ISO/IEC 15288: Systems engineering - System life cycle processes
- NIST: Recommended security controls - 800-53
- Successful Test Management: an integral approach - IRIS PINKSTER isbn-10 3-540-22822-5 Springer Berlin 2006
- ISO 15489-1: Information and documentation – records management – part 1: general
- ISO 14721:2003 (space information and data exchange systems – reference Model for an Open archival information system), better known as the Open Archival Information System (OIAS)
- ISO 15408: Information technology - Security technics - Evaluation criteria for IT security
- ISO IEC 60812: Analysis techniques for system reliability - Procedure for Failure Mode and Effects Analysis
- ARMA 8-2005: Retention Management for Records and Information
- RFC 3647: Internet X.509 Public Key Infrastructure - Certificate Policy and Certification Practices Framework

9. ANNEX 4 - Workshop members

Initiators / Sponsors

PROLOGISM *(CHAIRMAN)*

CS COMMUNICATION & SYSTEMES

LA BANQUE POSTALE

Participants

ARMA International	ASD	EISIS
Groupement des Cartes Bancaires	INFOCERT	NYSE EURONEXT Technology
RexConseil	THALES	VOLANS Informatica

Secretariat

afnor GROUPE

Association Française de NORmalisation (AFNOR), the French Member of CEN

PROLOGISM is a consulting company, expert in critical applications design and development. PROLOGISM has developed a broad expertise in architecture, design, development, re-engineering and optimization of critical applications especially in the "services" sector including financial transactions, on-line trading, e-services, etc.

CS is a Prime Contractor for the design, integration and operation of mission-critical systems and secured infrastructure. CS is involved in all aspects of a company's critical applications, infrastructure and processes.

La Banque Postale is an organisation from the banking sector that implements critical information systems for many business processes including on-line banking, multi-channel banking, trading platforms, financial transactions, etc.