BUSINESS REQUIREMENTS SPECIFICATION

The Business Requirements Specification formally documents the business needs, both existing needs and also the changes that are required in the status quo. It documents WHAT is required, and its primary audience is the designers, who will determine HOW these needs are to be met[1].

It is one of the two key outputs from the business analysis activity, and is a critical part of the overall project documentation. It does not stand alone, but must be distributed, understood, controlled and managed in the context of the Business Case and other project documentation.

It follows that, until the other project documentation exists, then the detailed specification activity can not proceed. Analysis requires a clear understanding of scope to commence, and will not be complete until tested against clearly identified project constraints (although constraints are not necessary to commence).And conversely, the project documentation, including the business case, is not complete until the Specification is complete. Only then can costings, design, benefits realisation planning and implementation planning proceed.

[1] The What and the How cascade down through to the lowest level. Thus the sub-system designers will then use the system design as a statement of WHAT is required, to generate a sub-system design, which specifies HOW the sub-systems needs will be met (eg: a functional specification for a sub-system). In turn, the component designers will use the sub-system design to generate component (module) designs, which specify HOW the component needs will be met (eg: the physical database structure, the usecases, and other aspects of the technical design). While the examples given relate to information-technology based systems, the same principle applies for other technologies, and indeed for that proportion of the overall system that will be practiced manually.

ABN: 97 081 830 499

GPO Box 2785
Canberra ACT 2601

fax: +61 3 6257 2081
www.blackcircle.com.au

THE BASIC FORMAT

*The ... System **shall** perform as documented in the following:*

- *Context Level Diagrams*
- *Data Flow Diagrams*
- *Work Flow Diagrams*
- *Data Dictionary & Entity-Relationship Diagram*
- *State Transition Diagrams*
- *Quality Matrix*

Attachments:

- *Business Case (inc. Design Constraints for people and technology)*
- *Benefits Realisation Plan (includes Implementation considerations)*
- *Current system Context Level Diagrams*
- *Current system Data Flow Diagrams*
- *Statement of Business Objectives*
- *Statement of Job Impact*
- *Glossary*

Requirements should be...

☑ Focused on the business need (WHAT), rather than physical implementation (HOW):

When these diagrams are generated through a business-focused communication process, then they will all be focused on the conceptual business need, and make no assumptions about implementation. Talking with stakeholders about HOW the process may be implemented inevitably leads the discussion down into the physical details of operational procedure or technology. This closes off options, and can lead to incomplete specification. By contrast, using the diagrams listed will avoid implementation detail, as they are technology-independent.

A second option for refocusing on WHAT rather than HOW is to ask stakeholders: “WHY do they do that”. This can elicit policy, legislative, cultural and procedural reasons or requirements to be met. However, it can backfire: stakeholders can hear the question as: “Why do YOU do that”, and become defensive—particularly if the only reason is that: “we have always done it like that”!

☑ Necessary:

If the requirement is in the diagrams, then clearly it is necessary to support the business objectives that will be addressed in the proposed business activity. To ensure relevance in the bigger context, check that there is a clear linkage between the project objective and the corporate objective(s) addressed by the proposal.

In doubt, ask: “what is the worst thing that could happen if this requirement was not included?” If there is no substantitive answer to this question, then consider removing it. In the case of business information systems, more often the answer will be that business benefits will be reduced, or support costs increased. In this case, the value of the benefits that would be foregone or the extra support costs, will give a ranking for the requirement.

Keeping the requirements firmly focused on the business rather than the physical at this stage guards against minor "nice-to-haves" creeping in and distorting the proposal.

☑ Unique:

Requirements must only be specified once. Duplication can lead to omission, or to errors creeping in as some are updated but not others.

If a requirement is spelt out in a diagram, do not duplicate it in text.

Lower levels inherit the requirements of the higher level, and therefore requirements should be specified at the higher level. Otherwise, the lower-level specification must be repeated for each element at that level.

These diagrams uniquely identify requirements: DataFlow, Flowchart and State Diagrams complement each other, and are not interchangable.

☑ Complete:

The diagrams listed above specify all necessary components of the conceptual model. The principles of dataflow diagramming ensure completeness:

- for any business cue (trigger, or event, documented in the state diagram), there must be a corresponding piece of business logic—a process.
- for any flow out of a process, there will be a corresponding flow in, and vice versa, thus ensuring no missing flows;
- the data dictionary will have entries for every piece of data travelling through the system.
- all data flowing into a process must be transformed, and accordingly the flowcharts will cover all necessary processing.

The Quality Matrix covers considerations such as safety, usability and reliability, where these have not already been resolved into data and function. Regulatory considerations are business requirements and are incorporated into the diagrams. Interface considerations are specified in the Context Level Diagram, as this documents all interfaces.

☑ Attainable:

The requirement must be achievable technically: that is, it must be within the capacity of current technology and scientific knowledge. That said, for many business information proposals, the technology **is** available, and in the 21st century it is a practical starting point to assume that. For in practice, it is necessary to draft a business model, and only then test it for technical feasibility, possibly iterating through this a couple of times.

The requirement must also be achievable within the project context, and particularly within the constraints (budget, time, resource and design). For most business information systems this is a far more stringent test of attainability. It is necessary to understand each business objective and the requirements to address it, and only then test these requirements for feasibility against the constraints, and again iterate through this a couple of times as both the benefits and the costs become clearer.

In the context of business information systems, when people question feasibility, they often really mean that it must be economically feasible on a specified platform (possibly a current system to be upgraded, or a pre-selected product). That is, they are not talking about technical feasibility at all, but about design and budget constraints. This often occurs when insufficient time has been spent understanding the benefits of the proposal, and the project therefore becomes cost-driven by default rather than outcomes-driven by intent.

☑ Verifiable:

Verification may occur through measurement, inspection or demonstration. The acceptance criteria must be specified. Tolerance must be clear; ambiguous or vague terms must not be used.

Separate specification of each requirement will assist verification, by allowing the testing process to be developed and implemented independently. If requirements are "bundled"(for example, with multiple data items travelling on a single dataflow) then they can only be tested together, and this may be impractical or inefficient (if one fails, then all fail, and must be retested in their entirity).

These diagrams all specify verifiable requirements: a dataflow will either exist, or not exist; a foreign key either will exist and thus enable navigation through the tables, or not; the quality attributes will be defined, and thus required, or not; State Diagrams can unambiguously specify response time and other interface requirements.

For interface requirements such as response time, specify the maximum tolerances that will deliver the business benefits, or if using standards, ensure that they specify reasonable tolerances. As tolerance tightens, so does cost increase. Equally, the probability of rejection during verification increases, with consequent slippage in schedule.

☑ Clear and Unambiguous:

The Specification must be consistent in style and layout, to minimise misinterpretation, and to assist readers. Use headings and formatting to structure the requirements and be consistent in their use. Numbering items rather than setting them as bullet points is less ambiguous for requirements; however bullet points may be appropriate to link items in background and explanatory material

Diagrams:

Diagrams are preferable, as they can be easier to read for visually-oriented readers, and generally take up less space that the equivalent text. However, they must be structured and unambiguous: the diagrams specified meet this requirement. All labels on DataFlow Diagrams are completely defined either in the Flowchart Diagram (for processes) or in the Data Dictionary (for DataFlows, DataStores and Entities).

Text:

Use abbreviations where they are short and therefore easier to read. Ensure that they are meaningful and acceptable to the business area. If used, ensure that they are all clearly defined in a glossary.

Be consistent with terms: do not change to new terminology whilst continuing to refer to the same thing or concept.

Do not use ambiguous terms or words such as: *better, and/or, etc, ought, may, but not limited to, support, assist, improve, easy-to-use, fast, rapid, adequate, user-friendly, phunky*. These terms are all unverifiable, and therefore not requirements. Some may be design goals, or strategic objectives: more likely, they reflect that insufficient time was spent on specification, and nobody really knows what the requirement is.

Long, complex sentences can be confusing or ambiguous. Keep sentences simple. The subject is the system component under discussion (system, sub-system, component process or datastore); the predicate is a unique and verifiable business requirement; and:

- For statements of fact, background or objectives, the verb is *will...*
- For a goal, the verb is *should (*it is not unambiguously verifiable),
- For a requirement, the verb is *shall,* and thus both verifiable, and enforceable.