Business Analyst Checklist V1

Contents

Overview: What is a Business Analyst? 3

Business Analyst Checklist 5

1. Work Activities 5
2. Skills / Attributes Required 5

Checklist for getting a JAD session started 6

1. Define the Project 6
2. Form the JAD Group 6
3. First JAD Meeting - Kick off Meeting 6
4. JAD Meetings - Planning, Analysis, Design Phase 6
5. JAD Meetings - Development, Execute, Finish Phase 6
6. Assign Duties 6

Plan Checklist 8

Effective Meetings 9

1. Before the Meeting 9
2. During the Meeting 9
3. After the Meeting 9
4. Five (more or less) Ps 9

Estimating Checklist 10

1. Things not to forget when estimating: 10

Project Document Checklist 11

Quality 12

1. Products (software and deliverables) 12
2. People 12
3. Process 12

Requirements Checklist 13

1. General 13
2. Individual Requirement 13
3. Business Requirements 13
4. Functional Requirements 14
5. Reliability 14
6. Performance 14
7. Security 14
8. Usability 14

Risk Checklist 15

1. Schedule 15
2. Organization and Management 15
3. Development environment 16
4. End-users 16
5. Customer 16
6. Contractors 16
7. Requirements 16
8. Product 16
9. External environment 17
10. Personnel 17
11. Design and Implementation 18
12. Process 18

Sponsor Checklist 19
Test Manager Checklist 20
Architect Checklist 21
Business Analysis Glossary - from A to Z 29

Overview: What is a Business Analyst?

"Today's Business Analyst may reside within any part of an organisation and this has a direct effect on the way they work and the deliverables they produce"

"Business Analysis is the process of understanding business change needs, assessing the impact of those changes, capturing, analysing and documenting requirements and then supporting the communication and delivery of those requirements with relevant parties."

A Bit of History

Requiring straightforward automation of repetitive administrative tasks and conversion from paper to electronic data storage, IT projects of the seventies and early eighties could not fail to be successful and reap financial rewards.

Systems Analysts took responsibility for documenting existing manual paper based processes, identifying problems and new business requirements, and then automating these processes through computerised systems. This provided significant savings in staff as well as improvements to customer service through access to electronic information in fractions of a second.

Throughout the late 1980's and 1990's, companies started to evolve their IT systems to take advantage of new technology as they attempted to make further savings or improvements in service. However, IT projects in this era continually failed. They either failed to deliver at all, or were delivered without providing any significant business benefits.

The reasons for failure were that projects became unfocussed, receiving (sometimes conflicting) demands from different business departments. Systems were developed with unrealistic business cases, without clear objectives, with unmanaged expectations of performance or merely to follow the 'emperors new clothes syndrome' of jumping on the latest technology bandwagon.

Business users became increasingly frustrated with the barriers that limit their ability to implement change promptly and effectively. As PC and server technology evolved, business users became wise to IT and started to purchase and build their own localised systems. This has left many companies in a position where as well as their existing 'legacy' systems, they have hundreds of different systems which often link in an uncontrolled fashion with no real documentation to explain the links.

The Business Analyst has Evolved

Throughout this period, the role of the Systems Analyst evolved into the Business Analyst. This role encompasses more than the ability to document processes and apply technological expertise.

While the Systems Analyst belonged to the IT department, Business Analysts can now be found within a number of places in organisation structures:

Within the IT department acting as a conduit to and from the business
Within individual business units with responsibility for identifying business needs
Within a change management department coordinating and managing change across the whole business
But wherever they sit, Business Analysts must be great communicators, tactful diplomats, problem solvers, thinkers and analysers - with the ability to understand and respond to user needs in rapidly changing business environments.

We define the purpose of the role of the Business Analyst as being ultimately responsible for ensuring that organisations get the most from their limited IT and change management resource.

Business Analysts are responsible for identifying change needs, assessing the impact of the change, capturing and documenting requirements and then ensuring that those requirements are delivered by IT whilst supporting the

business through the implementation process. Business Analysts should not just write specifications and then leave them to be delivered. The development lifecycle is an iterative one and the Business Analyst must be involved from initial concept through to final implementation.

Business Analysts are likely to be the key change facilitators within your organisation. They must deliver effective solutions which provide tangible business benefits usually within short timescales.

BA Job Description: a job description which may be tailored for use by a Business Analyst without any people management responsibilities.

Requirements document template: can be used as the template for capturing requirements and customised as required.

The Training Problem

Business Analyst training has not moved with the pace of business change nor with advances in information technology. Many of our competitors' training courses were developed in the eighties and have a strong bias on 'methods' or 'approach'. Such courses are geared towards a 'systems analysis and design' approach and fail to address the issues that the Business Analyst encounters within modern organisations and projects.

Developed over the last five years and under continuous enhancement, our training reflects the current and future demands of business change projects. Our mission is to ensure delivery of real business benefits through effective analysis.

Click here for more details of our Business Analyst Training courses.

The Recruitment Problem

We tend to find that traditional recruitment agencies have little concept of the skills required for the business analyst role. Clients application of 'quotas' and recent use of the web to find suitable candidates often means that the best applicants' CV's are never even presented to the client.

Employing good Business Analysts makes a real difference to the success of your projects

Business Analyst Checklist

1. Work Activities

- Collecting, understanding, and transmitting the business requirements for the project, and translating these into functional specifications and detailed test plans.
- Analyse and document business processes.
- Prepare functional specifications
- Document workflows and results of business analysis and obtain sign-off from client on the specifications.
- To provide the link between the customer, development team and any third party regarding software functionality, throughout the development lifecycle.
- To design and execute the test scenarios and test scripts.
- Assist in preparation of user and system test plans
- Day to day management of change requests in relation to the project plans to ensure agreed deadlines are met.
- Weekly reports to be produced for the project manager showing progress against outstanding milestones, status, resource requirements, issues, risks and dependencies.

2. Skills / Attributes Required

- Demonstrable evidence of analysing and documenting complex business processes.
- Demonstrable experience writing requirements specifications for Information Systems.
- A proven track record in Software Development
- End to end experience of the project lifecycle
- Proven experience interacting directly with end users.
- Results orientated with good communication and interpersonal skills

Checklist for getting a JAD session started

1. Define the Project

- The JAD Project Leader meets with the Project Sponsor to complete a JAD Project Charter.

2. Form the JAD Group

- The Project Leader and Project Sponsor form the JAD Group making sure you have all affected areas represented. You will need a Project Sponsor, Project Leader, Business Users and Systems Analysts. A JAD Group should have 8 or fewer total members. It is hard to be effective with more than 15 members.

3. First JAD Meeting - Kick off Meeting

Your first JAD meeting may have the following agenda items:

- Share problem definition and overall goal. Get consensus on these two items.
- Train each member of the new group on what a JAD Group is so they will understand the purpose, the roles and how a JAD works.
- Establish JAD Group expectations/responsibilities.
- Determine meeting frequency, time and place.
- Determine roles - Project Sponsor, Project Leader, Record Keeper, Timekeeper, Clients.
- Continue holding JAD meetings approximately every week or every other week until you have reached consensus on a design.

4. JAD Meetings - Planning, Analysis, Design Phase

- Review the current process - map it out
- Identify Problems/Challenges in the current process
- Brainstorm solutions for those problems and challenges
- Benchmark other organizations for possible solutions
- Consider Buy vs. Build
- Survey your customers for problems and ideas
- Evaluate list of generated ideas
- Determine your course of action - tasks to be accomplished
- Develop your timebox - list of tasks, who is assigned and when each task is due.
- Present the Project Design to the Project Sponsor and Representative Customers and Get the Thumbs Up
- Communicate, Communicate, Communicate

5. JAD Meetings - Development, Execute, Finish Phase

- Meet every 2 weeks to make sure the development stays on track Agenda - how did we do on our goals?
- Discuss problems and challenges
- Make decisions jointly
- Set goals for next meeting
- Assign tasks

6. Assign Duties

- Assign as many of the project duties as possible to members of the JAD - this helps build buy in and a feeling of ownership towards the project.

Plan Checklist

- The plan is based on the proper template.
- The plan document complies with documentation standards.
- All assumptions on which the plan is based are adequately identified and described.
- All work defined in the complete project plan is necessary to satisfy the charter, control risks, and/or maximize assets.
- All available resources are adequately identified and all resources identified as being
- available are actually available.
- The available resources are adequately and appropriately allocated to the defined work (i.e., not over or under-utilized).
- All derived/induced risks and assets have been identified and accounted for in the complete project plan.
- Uncertainties (“TBD”s) in the project plan are both obviously identified and appropriate.
- Where appropriate, all project plan components (deliverables, practices, resource allocations, etc.) are explicitly justified in light of the charter, and Risk assessments.

Effective Meetings

1. Before the Meeting

- Be clear on purpose and aims.
- Create the agenda.
- Schedule the meeting.
- Ensure that agenda is posted and sent out.
- Ensure that appropriate supporting information is circulated in time to be useful.
- Ensure that room arrangements (including refreshments) are made.
- Arrange for recorder (and supplies such as flip chart, markers, etc.).

2. During the Meeting

- Start meeting on time.
- Ensure quorum (if required).
- Review agenda.
- Keep discussion focused on agenda items.
- Encourage full participation.
- Help group come to decisions.
- Summarize decisions.
- Agree on action plan: point person and what needs to be done by whom by when.
- Draft agenda for next meeting(s).
- Evaluate the meeting.

3. After the Meeting

- Ensure that minutes are produced and promptly distributed, including to guests.
- Ensure that agenda, minutes and meeting and supporting documents are kept together and archived as required.
- Check to ensure that action is taking place as agreed.

4. Five (more or less) Ps

- People
- Place
- Purpose (Subject) (Prep-Work)
- Process (Agenda)
- Payoff (What is at stake) (Parked list to stay on track)

Estimating Checklist

1. Things not to forget when estimating:

- Ramp up time for new team members
- Training of the team
- Management meetings
- Cutover/deployment trials
- Data conversion
- Installation
- Customization
- Requirements clarifications
- Maintaining configuration control systems
- Maintaining automated scripts
- Creation of test data
- Technical reviews
- Change requests
- Change request analysis
- Maintenance work on old systems
- Defect fixing
- Performance tuning
- Learning new tools or technology
- Management of defects/testing
- Answering questions
- User documentation
- Demonstrations
- Prototypes
- Reviews - plans/designs/requirements/estimates/code/scripts etc.

Project Document Checklist

- The project brief/definition is based on the charter template.
- The charter document complies with documentation standards.
- The sponsor that is chartering the project is clearly identified.
- All stakeholders in the project have been identified. This includes anyone who is responsible for seeing that the project is carried out successfully.
- All identified stakeholders are really responsible, and they are being delegated authority, in the project.
- All true completion criteria have been clearly identified.
- Every completion criterion is really applicable to the project.
- The completion criteria are as objectively measurable as possible.
- The completion criteria do not confuse the problem with the solution. In other words, the completion criteria are problems to be solved, not solutions to be implemented.
- All appropriate criteria for cancelling the project are included.
- All resources being made available are clearly identified.
- All significant constraints are clearly identified.
- All stated constraints are really constraints on this project.
- All assumptions are clearly identified and relevant.

Quality

- It is often a struggle to decide what should go into a quality plan. It becomes easier if you break it down into the main areas (the 3Ps):

1. Products (software and deliverables)

- How will we test them?
- How will we know the meet the requirements or are complete?
- How will we inspect them (eg peer review)?
- How will we number them (eg version or configuration control)?
- How will we store and distribute them?

2. People

- What is expected of them (start time etc.)?
- How will they record their time?
- How will they do their work (document standards, filing standards)?

3. Process

- What processes need to be adhered to (change, risk, issue etc.)?
- How will we capture any deviation to a process?
- What role does an individual play in a process?

Requirements Checklist

Table of Contents

General
Individual Requirement
Business Requirements
Functional Requirements
Reliability
Performance
Security
Usability

1. General

- Are the requirements complete?
- Are all requirements uniquely identifiable?
- Are the requirements clearly and appropriately prioritised?
- Are the requirements consistent? (i.e., no internal contradictions)
- Does the set of requirements adequately address all appropriate exception conditions?
- Does the set of requirements adequately address boundary conditions?
- Are the requirements feasible? (i.e., a solution to the set of requirements exists)
- Can the requirements be implemented within known constraints?
- Are the requirements sufficient? (i.e., they could be sent to a reputable development organization and have a reasonable probability of producing the product that was desired)
- Are inverse requirements explicitly stated?
- Are these the simplest set of requirements that meets the stakeholders needs?
- Are all cross-references to other requirements correct?
- Have functional and non-functional requirements been considered?

2. Individual Requirement

- Is the requirement precise and unambiguous?
- Is the requirement stated in as simple or atomic a form as possible?
- Is the requirement testable/verifiable?
- Is the requirement correct?
- Is the requirement in scope? (i.e., the system will be considered incomplete if even one requirement is left out)
- Is the requirement as modifiable as possible?
- Is the requirement written in the customer's language, using the customer's terminology?
- Is the requirement acceptable to all stakeholders?
- Is the requirement a statement of stakeholder need, not a solution?
- If appropriate, is the requirement traceable?
- Is the requirement necessary?
- Are all missing items or unresolved issues identified with a TBD, an owner, and a time-line for closing it?

3. Business Requirements

- Are the high-level business objectives described?

- Is a system perspective used when appropriate? (i.e., business requirements often do not deal exclusively with software behaviour, but with the interactions of software with other parts of a larger system)
- Are the requirements understandable by all stakeholders?
- Is the value to the business identified? (cost savings, reduced inventory, etc.)
- Is the value to the customer identified? (new features, improved usability, etc.)
- If appropriate, are the business opportunities the requirements support outlined?
- Are the details of how the system will meet objectives avoided?
- Does this requirement answer the question 'Why are we doing this software'?
- Is this requirement a product or business vision?
- Is this requirement a product or business goal?
- Is this requirement a customer goal?
- Is this requirement a system objective?
- Does this requirement describe a product charter?

4. Functional Requirements

- Does the set of functional requirements meet the needs outlined by business requirements? (e.g. complete, sufficient, etc.)
- Is the relation between functional and the non-functional requirements clear?
- Is the relation between functional requirements and any user interface designs clear?
- Do the functional requirements convey enough knowledge to allow design activities to begin or continue?
- Do the functional requirements avoid design and construction details?
- Are the functional requirements at an appropriate level of precision? (e.g., with an evolutionary prototyping lifecycle, functional requirements should NOT be extremely precise; details will be worked out as design and construction progress)
- Are appropriate mechanisms used to capture the functional requirements? (e.g., user interface prototypes and screen mockups are often better than text)

5. Reliability

- Are the reliability requirements specified?
- Are the availability (up time) requirements specified?
- Are the serviceability requirements specified?
- Are the robustness requirements specified?

6. Performance

- Are the response time or latency requirements specified?
- Are the throughput requirements specified?
- Are the data volume requirements specified? (input, stored, output)
- Are the peak or short-term load requirements specified?

7. Security

- Are the security requirements specified?
- Are the safety requirements specified?

8. Usability

- Are the usability requirements specified?
- Are the internationalization/localization requirements specified?
- Are the look and feel requirements specified? (e.g. colour schemes, standards, etc.)

Risk Checklist

Table of Contents

Schedule
Organization and Management
Development environment
End-users
Customer
Contractors
Requirements
Product
External environment
Personnel
Design and Implementation
Process

1. Schedule

- Schedule, resources, and product definition have all been dictated by the customer or upper management and are not in balance
- Schedule is optimistic, "best case," rather than realistic, "expected case"
- Schedule omits necessary tasks
- Schedule was based on the use of specific team members, but those team members were not available
- Cannot build a product of the size specified in the time allocated
- Product is larger than estimated (in lines of code, function points, or percentage of previous project's size)
- Effort is greater than estimated (per line of code, function point, module, etc.)
- Re-estimation in response to schedule slips is overly optimistic or ignores project history
- Excessive schedule pressure reduces productivity
- Target date is moved up with no corresponding adjustment to the product scope or available resources
- A delay in one task causes cascading delays in dependent tasks
- Unfamiliar areas of the product take more time than expected to design and implement

2. Organization and Management

- Project lacks an effective top-management sponsor
- Project languishes too long in fuzzy front end
- Layoffs and cutbacks reduce team's capacity
- Management or marketing insists on technical decisions that lengthen the schedule
- Inefficient team structure reduces productivity
- Management review/decision cycle is slower than expected
- Budget cuts upset project plans
- Management makes decisions that reduce the development team's motivation
- Non-technical third-party tasks take longer than expected (budget approval, equipment purchase approval, legal reviews, security clearances, etc.)
- Planning is too poor to support the desired development speed
- Project plans are abandoned under pressure, resulting in chaotic, inefficient development
- Management places more emphasis on heroics than accurate status reporting, which undercuts its ability to detect and correct problems

3. Development environment

- Facilities are not available on time
- Facilities are available but inadequate (e.g., no phones, network wiring, furniture, office supplies, etc.)
- Facilities are crowded, noisy, or disruptive
- Development tools are not in place by the desired time
- Development tools do not work as expected; developers need time to create workarounds or to switch to new tools
- Development tools are not chosen based on their technical merits, and do not provide the planned productivity

4. End-users

- End-user insists on new requirements
- End-user ultimately finds product to be unsatisfactory, requiring redesign and rework
- End-user does not buy into the project and consequently does not provide needed support
- End-user input is not solicited, so product ultimately fails to meet user expectations and must be reworked

5. Customer

- Customer insists on new requirements
- Customer review/decision cycles for plans, prototypes, and specifications are slower than expected
- Customer will not participate in review cycles for plans, prototypes, and specifications or is incapable of doing so—resulting in unstable requirements and time-consuming changes
- Customer communication time (e.g., time to answer requirements clarification questions) is slower than expected
- Customer insists on technical decisions that lengthen the schedule
- Customer micro-manages the development process, resulting in slower progress than planned
- Customer-furnished components are a poor match for the product under development, resulting in extra design and integration work
- Customer-furnished components are poor quality, resulting in extra testing, design, and integration work and in extra customer-relationship management
- Customer-mandated support tools and environments are incompatible, have poor performance, or have inadequate functionality, resulting in reduced productivity
- Customer will not accept the software as delivered even though it meets all specifications
- Customer has expectations for development speed that developers cannot meet

6. Contractors

- Contractor does not deliver components when promised
- Contractor delivers components of unacceptably low quality, and time must be added to improve quality
- Contractor does not buy into the project and consequently does not provide the level of performance needed

7. Requirements

- Requirements have been baselined but continue to change
- Requirements are poorly defined, and further definition expands the scope of the project
- Additional requirements are added
- Vaguely specified areas of the product are more time-consuming than expected

8. Product

- Error-prone modules require more testing, design, and implementation work than expected
- Unacceptably low quality requires more testing, design, and implementation work to correct than expected
- Development of the wrong software functions requires redesign and implementation

- Development of the wrong user interface results in redesign and implementation
- Development of extra software functions that are not required (gold plating) extends the schedule
- Meeting product's size or speed constraints requires more time than expected, including time for redesign and re-implementation
- Strict requirements for compatibility with existing system require more testing, design, and implementation than expected
- Requirements for interfacing with other systems, other complex systems, or other systems that are not under the team's control result in unforeseen design, implementation, and testing
- Pushing the computer science state-of-the-art in one or more areas lengthens the schedule unpredictably
- Requirement to operate under multiple operating systems takes longer to satisfy than expected
- Operation in an unfamiliar or unproved software environment causes unforeseen problems
- Operation in an unfamiliar or unproved hardware environment causes unforeseen problems
- Development of a kind of component that is brand new to the organization takes longer than expected
- Dependency on a technology that is still under development lengthens the schedule

9. External environment

- Product depends on government regulations, which change unexpectedly
- Product depends on draft technical standards, which change unexpectedly

10. Personnel

- Hiring takes longer than expected
- Task prerequisites (e.g., training, completion of other projects, acquisition of work permit) cannot be completed on time
- Poor relationships between developers and management slow decision making and follow through
- Team members do not buy into the project and consequently does not provide the level of performance needed
- Low motivation and morale reduce productivity
- Lack of needed specialization increases defects and rework
- Personnel need extra time to learn unfamiliar software tools or environment
- Personnel need extra time to learn unfamiliar hardware environment
- Personnel need extra time to learn unfamiliar programming language
- Contract personnel leave before project is complete
- Permanent employees leave before project is complete
- New development personnel are added late in the project, and additional training and communications overhead reduces existing team members' effectiveness
- Team members do not work together efficiently
- Conflicts between team members result in poor communication, poor designs, interface errors, and extra rework
- Problem team members are not removed from the team, damaging overall team motivation
- The personnel most qualified to work on the project are not available for the project
- The personnel most qualified to work on the project are available for the project but are not used for political or other reasons
- Personnel with critical skills needed for the project cannot be found
- Key personnel are available only part time
- Not enough personnel are available for the project
- People's assignments do not match their strengths
- Personnel work slower than expected

- Sabotage by project management results in inefficient scheduling and ineffective planning
- Sabotage by technical personnel results in lost work or poor quality and requires rework

11. Design and Implementation

- Overly simple design fails to address major issues and leads to redesign and re-implementation
- Overly complicated design requires unnecessary and unproductive implementationoverhead
- Inappropriate design leads to redesign and re-implementation
- Use of unfamiliar methodology results in extra training time and in rework to fix first-time misuses of the methodology
- Product is implemented in a low level language (e.g., assembler), and productivity is lower than expected
- Necessary functionality cannot be implemented using the selected code or class libraries; developers must switch to new libraries or custom build the necessary functionality
- Code or class libraries have poor quality, causing extra testing, defect correction, and rework
- Schedule savings from productivity enhancing tools are overestimated
- Components developed separately cannot be integrated easily, requiring redesign and rework

12. Process

- Amount of paperwork results in slower progress than expected
- Inaccurate progress tracking results in not knowing the project is behind schedule until late in the project
- Upstream quality-assurance activities are shortchanged, resulting in time-consuming rework downstream
- Inaccurate quality tracking results in not knowing about quality problems that affect the schedule until late in the project
- Too little formality (lack of adherence to software policies and standards) results in miscommunications, quality problems, and rework
- Too much formality (bureaucratic adherence to software policies and standards) results in unnecessary, time-consuming overhead
- Management-level progress reporting takes more developer time than expected
- Software project risk management takes more time than expected

Sponsor Checklist

- Define overall vision, direction, scope and constraints for the project manager.
- Ensure project charter is created and accepted by all stakeholders.
- Ensure project charter is maintained and updated throughout the life of the project.
- Manage organizational risks of the project.
- Review status reports.
- Review project plans and deliverables, as appropriate, on a regular basis.
- Meet regularly with project business manager to discuss project.
- Help resolve any disputes between internal project team and external stakeholders.
- Allocate staffing to project as appropriate.
- Assist resolving any project disputes as appropriate.
- Approve changes in scope or budget

Test Manager Checklist

The role of the software test manager or test lead is to effectively lead the testing team. To fulfill this role, the lead must understand the discipline of testing and how to effectively implement a testing process while fulfilling the traditional leadership roles of a manager by leading his team. The manager must manage and implement or maintain an effective testing process. That involves creating a test infrastructure that supports robust communication and a cost-effective testing framework.

What the test manager is responsible for:

- Defining and implementing the test approach.
- Defining the scope of testing.
- Deploying and managing the appropriate testing framework to meet the test approach.
- Implementing and evolving appropriate measurements and metrics. To be applied against the product under test.
- To manage the testing team and ensure all resources are used effectively.
- Planning, deploying and managing the testing effort for any given engagement/release.
- Managing and growing testing assets required for meeting the test approach
- Testing estimates
- Testing processes
- Effective test metric and progress reporting
- Identifying risks and issues associated with the testing approach

The test manager or lead must understand how testing fits into the project structure. It is the test lead's job to communicate and implement effective managerial and testing techniques to support the project.

The definition of scope will change as you move through the various stages of testing. The key thing is to make sure the testing team clearly understands what is being tested and what is not being tested for the current release and testing procedures to employ.

Architect Checklist

- Identify design risks.
- Management for design issues.
- Provide guidance to team members
- Provides work breakdown for design related tasks.
- Assist in estimation activities as needed
- Participate in the regular Change meetings as needed.
- Be the primary technical resource for design issues related to the project.
- Create and maintain the architecture specification.
- Lead high level design activities.
- Ensure creation and maintenance of all design artefacts.
- Oversee and assist with low level design activities.
- Work with the BA to ensure the requirements are sufficiently complete to support design.
- Work with the SWE to ensure feasibility of designs.
- Attempt to participate in all design discussions so there is one person who has the entire system design in their head (as much as that is possible).
- Ensure design documents are reasonably up to date with any changes introduced in release, or CR entered for future design document update.
- Collect and record design lessons learnt
- Design and document all required environments for successful solution development
- Ensure connectivity and set up of any other system the solution interfaces with (for End to End testing)
- Design and set up the appropriate security model for the solution in all environments
- Develop the package technical implementation plan

WBS Dictionary

1. General Data

Project name:

Project ID:

Project manager:

Project key stakeholders:

Document created (date):

Document created by:

Document Version:

This version created (date):

This version created by:

Work Breakdown Structure WBS

Work Breakdown Structure WBS

Instructions:

This template presents a standard Work Breakdown Structure (WBS) design for planning and monitoring IT projects at NCDOT. It provides a framework for developing a plan/schedule for any project.

To create a schedule in the selected project management tool, refer to the WBS template and perform the following steps:

1. Set up the Project Initiation, Planning, Execution, Controlling and Closing Phases as the 2nd Level of the WBS.

   Note: the Project Definition is the 1st Level of the WBS.

2. Review the 3rd Level WBS elements listed under each project phase and:

   a. Delete those not relevant to your project.

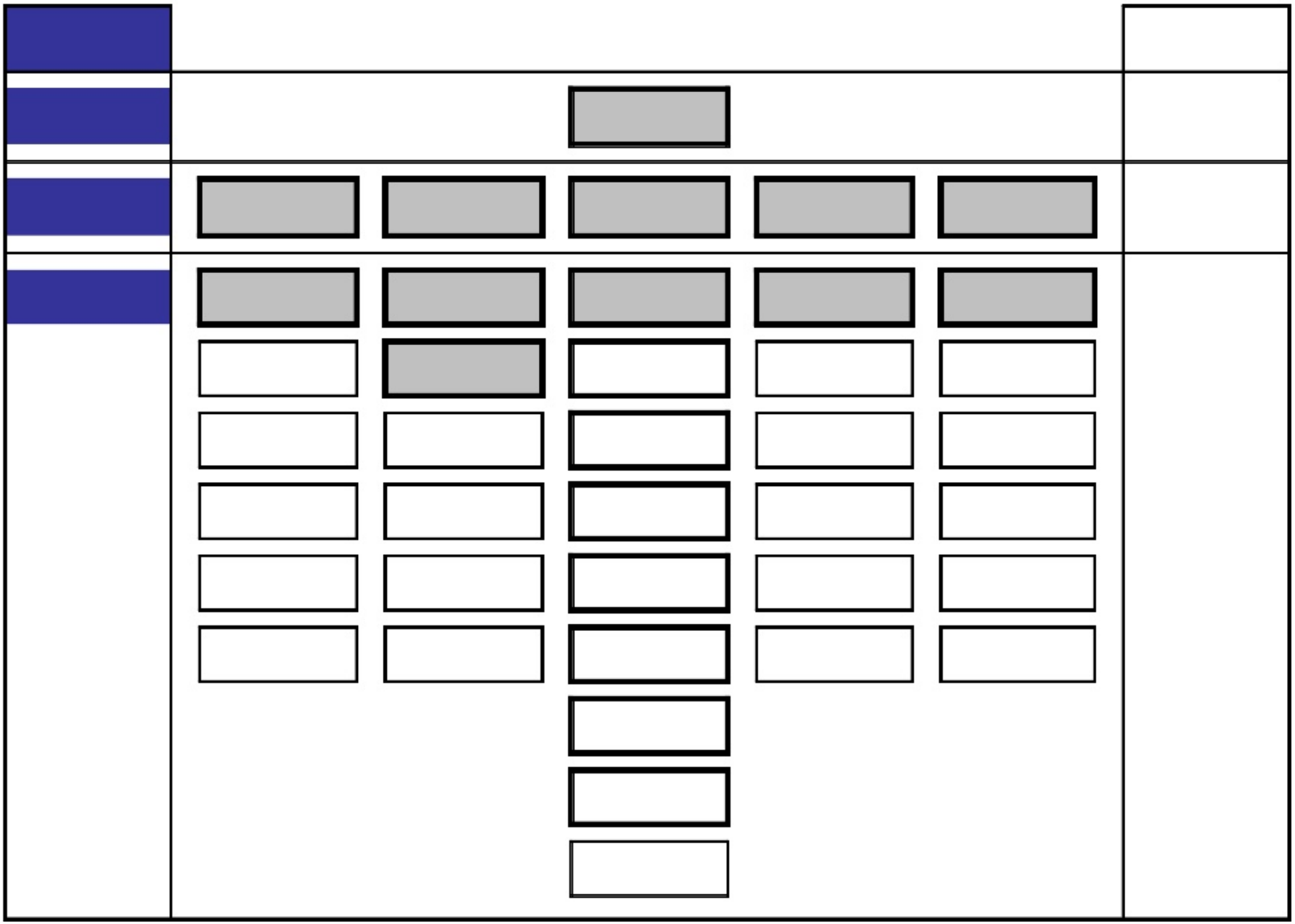
   b. Add new elements (add appropriate number of "Other" boxes and rename)

3. Incorporate WBS into project management tool.

You must have "Tools/Options/View/Comment Indicator Only" checked in order to see on -line help.

PROJECT INFORMATION	
Project Description	
Project Definition	
IT Project Manager	

Project System of Transportation

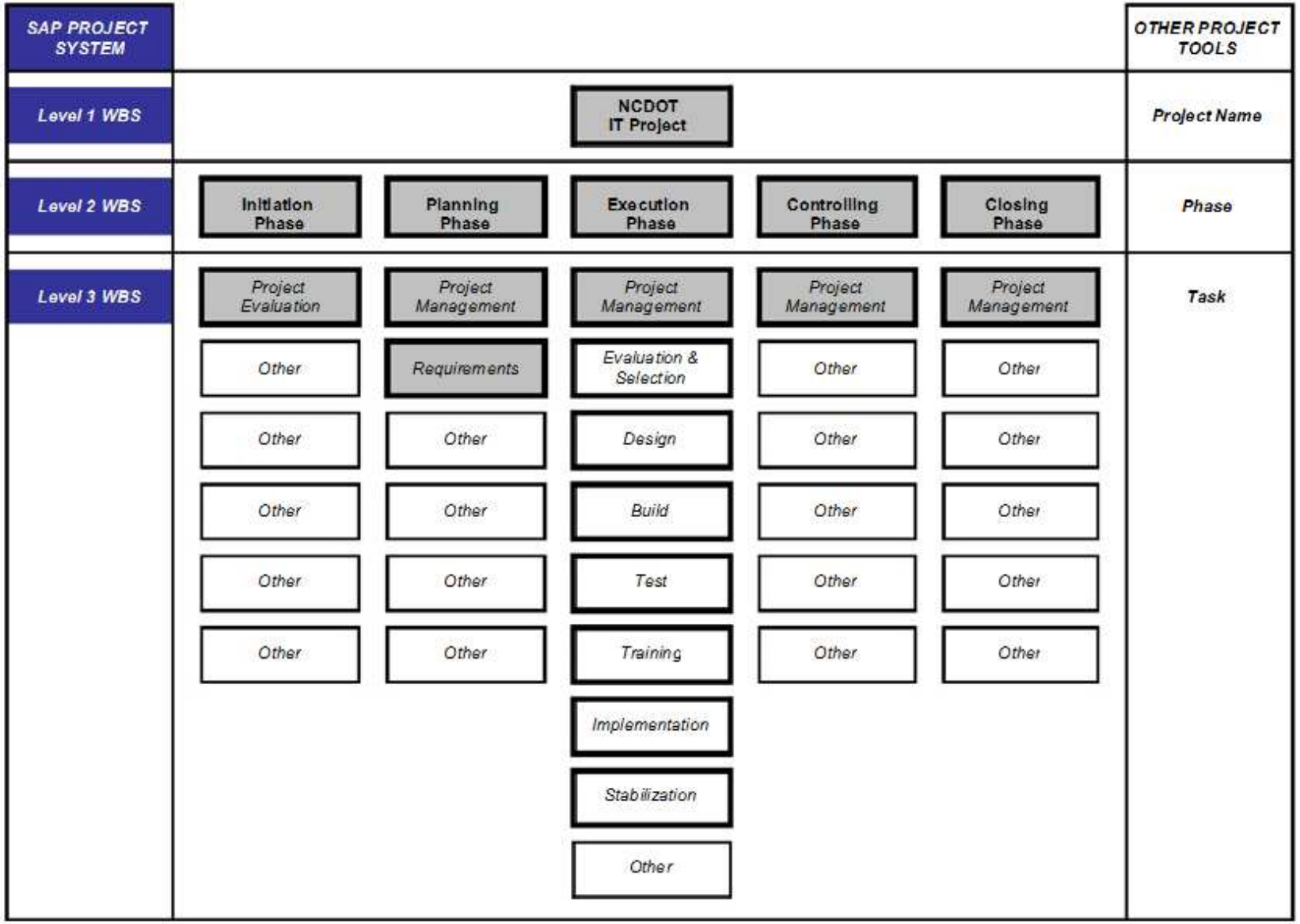


Business Analyst Checklist V1

Information Technology - Project System

WBS Template

North Carolina Department of Transportation



Business Analysis Glossary - from A to Z

A

Active Listening
Active Listening is a method used to listen and respond to others in a structured and deliberate way. It requires a listener to understand and actively evaluate what he or she heard.

Activity Diagram
An activity diagram is a UML diagram that is used to model a process. It models the actions (or behaviors) performed by the components of a business process or IT system, the order in which the actions take place, and the conditions that coordinate the actions in a specific order. Activity diagrams use swim lanes to group actions together. Actions can be grouped by the actor performing the action or by the distinct business process or system that is performing the action.

Alternative Flow
An alternate flow describes a use case scenario other than the basic flow that results in a user completing his or her goal. It is often considered to be an optional flow and implies that the user has chosen to take an alternative path through the system.

B

Business Analysis Planning and Monitoring (BABOK Knowledge Area)
A BABOK 2.0 Knowledge Area that describes how a business analyst determines which activities will be needed to complete the business analysis effort. The tasks within this knowledge area govern the business analysis tasks in all of the other knowledge areas.

Burndown Chart
A Burndown Chart is a tool used by multiple software engineering methods to track the progress of work completed. It compares the amount of work remaining (typically measured along the vertical axis) against time (measured along the horizontal axis). The burndown chart gives a quick view of the amount of work that is completed over time.

Business Entity Model
A business entity model is a logical model that documents the entities, or things, that a business or business process uses and interacts with in order to accomplish its business activities and goals. In addition to documenting entities, a business entity model may capture the attributes of an entity, relationships between entities, and cardinality information. Many business entity models are created in the form of a UML class diagram.

C

CBAP
See Certified Business Analysis Professional

Certified Business Analysis Professional
The Certified Business Analysis Professional certification (CBAP certification) is the designation given to those professionals who sit for and pass the CBAP exam. For this reason, the term CBAP is often used as a shorthand term to refer to the CBAP exam itself. More info on CBAP - Certified Business Analysis Professional

Class Diagram
A class diagram is a UML diagram that describes the structure of a system by showing the classes of a system, the attributes and operations that belong to each class, and the relationships between the classes.

Concentration Ratio
Concentration Ratio (CR) is a measurement used to understand the level of competition that exists within a market or industry in which a company operates.

Context Diagram
A context diagram is a special form of a data flow diagram that represents an entire system as a single process and highlights the interactions between the system being analyzed and other systems or people that interact with it.

Convergent Thinking
Convergent thinking is the process of focusing on a few sets of ideas and evaluating them based on selection criteria in order to narrow down the available options.

Cost Benefit Analysis
Cost Benefit Analysis is a technique used to determine if the financial benefits of a project outweigh the associated cost of undertaking the project in the first place. For a short term project where the benefit may be an immediate one-time cash windfall this may be as simple as subtracting the total of all project costs from the total of all project benefits. If the total is positive, then the project may be worth completing.

CRUD
CRUD stands for: Create, Read, Update, Delete. These are the four basic functions that can be performed when working with data in a persistent storage.

D

Data Flow Diagram
A data flow diagram models the system as a network of functional processes and its data. It documents the system's processes, data stores, flows which carry data, and terminators which are the external entities with which the system communicates.

Decision Table
A decision table is an unambiguous and compact technique for modeling complicated logic using several sets of conditions in a tabular format. It is often used to model logic that may otherwise require many sentences or paragraphs to convey.

Decision Tree
A decision tree graphically represents a series of decision points with branching occurring at each decision point forming a treelike structure. A decision tree maps out each possible outcome and will often also include the probability of each outcome.

Discount Rate
The discount rate is the percentage rate used to reduce future cash flow values for each year in the future that they occur. This is necessary to determine what the comparable cash flow amount would be in present terms.

Divergent Thinking
Divergent thinking is the process of generating many ideas that branch out from an original topic or concept.

E

Elicitation (BABOK Knowledge Area)
A BABOK 2.0 Knowledge Area that describes the steps required to elicit requirements from stakeholders. It includes preparing for elicitation by identifying a combination of techniques that will be used, conducting the elictation using the idetified techniques, documenting the elicitation results, and confirming what has been documented.

Enterprise Analysis (BABOK Knowledge Area)
A BABOK 2.0 Knowledge Area that describes the business analysis activities required to compare the needs of the business against the current capabilities of the business and identify opportunities for improvement. Then, based on this information, the analyst can determine which solutions should be selected to resolve the issue.
Stakeholder Analysis.

Entity Relationship Diagram
An entity-relationship diagram models the relationships between entities in a database. Standard symbols are used to represent different types of information. The conventional notation uses rectangles to represent entities (nouns), diamonds to represent relationships (verbs) and ovals to represents attributes of entities. Other notations are sometimes used.

Exception Flow
A use case exception flow is an unintended path through the system usually as a result of missing information or system availability problems. Exception flows represent an undesirable path to the user. However, even though the exception flow has occurred the system will ideally react in a way that recovers the flow and provide some useful information to the user.

F

Fishbone Diagram
A fishbone diagram is a problem-analysis tool that derives it's name from it's shape which resembles the skeleton of a fish. Developed by Dr. Kaoru Ishikawa, a Japanese quality control statistician, the fishbone diagram is a systematic way of looking at an effect and identifying and capturing the causes that contribute and result in that particular effect. For this reason, it is sometimes referred to as a cause and effect diagram.

Financial Ratio Analysis
Financial Ratio Analysis is the evaluation and interpretation of a company's financial data using standard financial ratios or accounting ratios to determine a company's financial state or condition. A financial ratio or accounting ratio is a ratio of two values that are taken for a company financial statements (Balance Sheet, Income Statement, Statement of CashFlows, Statement of Retained Earnings).

G

H

Herfindahl Hirschman Index
The Herfindahl Hirschman Index (HHI) is a measurement used to understand the level of competition that exists within a market or industry, as well as give an indication of how the distribution of market share occurs across the companies included in the index.

HTML
See HyperText Markup Language

HyperText Markup Language
HyperText Markup Language or HTML is used to define the structure of webpages. Markup languages describe annotations that are added to any document that are distinguishable from the original text of the document. In the case of HTML, these annotations are HTML tags which are used to define the structure of a webpage such as headings, paragraphs, lists, tables, data, quotes, and more.

I

J

Joint Application Development
Joint Application Development is a requirements-definition and software system design methodology in which stakeholders, subject matter experts (SME), end-users, business analysts, software architects and developers attend collaborative workshops (called JAD sessions) to work out a system's details.

K

Knowledge Areas
Categories of related information and tasks that a business analyst must understand and apply. This term is most often used by the BABOK (Business Analysis Body of Knowledge).

L

M

Management By Walking Around
Management By Walking Around (MBWA) is a popular management technique used by top-level managers in traditional brick and mortar businesses where managers walk around and observe the work, culture, atmosphere, and problems that may exist.

Model-Based-Management
Model-Based Management refers to the activity of managing and making informed decision regarding the future direction of a business, process, or system(s) based on information gleaned and understood from models that document the current state.

Model-View-Controller
Model-View-Controller, or MVC, is a design and architectural pattern used to ensure that the modeling of the domain, the presentation information, and the actions taken based on user input are loosely coupled and maintained as separate classes.

N

Non-Functional Requirement
Non-functional requirements are characteristics of a system or solution which describe non-behavioral characteristics or qualities of a system. Non Functional Requirements have also been called the 'ilities': usability, reliability, interoperability, scalability, extensibility, etc. Non-functional requirements are also commonly referred to as quality of service (QoS) requirements or service-level requirements. More info on non-functional requirements.

O

P

PDCA Method
A 4-step, iterative method commonly used for Business Process Improvement. PDCA stands for Plan, Do, Check, Act. It is used to create a feedback loop based on measurable results and make incremental changes and improvements over time.

Primary Actor
Primary actors are people, or at times even other systems, that require the assistance of the system under consideration to achieve their goal. They initiate the use cases of the system (business processes or application functionality). A use case within the system may have more than one primary actor, since more than one type of role may initiate the processes or functionality of the system.

Problem Domain
Problem Domain describes the area undergoing analysis, and includes everything that needs to be understood in order to achieve the goal of the project. This may includes all inputs and outputs of a process, any related systems, and internal and external project stakeholders.

Pseudocode
Pseudocode is a notation that combines some of the structure of a programming language, such as IF-ELSE and DO WHILE constructs, with a natural language, such as plain English. This allows writers of specification to eliminate a lot of the ambiguity that typically arises when trying to describe logic and computations using strictly a natural language.

Q

Quality Assurance
Quality Assurance is about Process. It describes the proactive method of establishing a process that is capable of producing a product or deliverable that is error or defect free.

Quality Control
Quality Control is about Products or Deliverables. It describes checking a final product or deliverable to ensure that it is defect or error free and meets specifications.

R

Requirement
A documented representation of a condition or capability. Specifically, one that is needed by a stakeholder to solve a problem or achieve an objective, or one that must be met or possessed by a solution to satisfy a contract, standard, specification.

Requirements Management and Communication (BABOK Knowledge Area)
A BABOK 2.0 knowledge area that describes what is involved in managing and articulating requirements to a wide variety of stakeholders. It includes understanding the link between business or project objectives and the specific requirements that comes from them such that any change or clarification in the objectives will result in a revised set of requirements that reflect t he business need.

Requirements Traceability Matrix
A Requirements Traceability Matrix is a tabular format that provides the ability to follow and audit the life of a requirement, in both a forward and backward direction: from its origins, through its realization in the design and functional specifications, to i ts eventual development and deployment and use, and through subsequent rounds of modification and refinement.

Role
A role describes a related set of activities that a single person may regularly undertake in order to partially or fully comp lete a process or goal. A role is different than a job title. Roles, reporting structures, and other parameters may all be used in conjunction to define a job title.

RuleSpeak
RuleSpeak is a set of guidelines for expressing business rules using a natural language (such as English). Rulespeak is not a language or syntax itself but rather a set of guidelines to facilitate the creation of business rules that are concise, consi stent, and less ambiguous. RuleSpeak is fully consistent with the OMG's SBVR standard.

S

Secondary Actor
A secondary actor is a person, business processes, or applications that provides a specific result or information to a use case in order for the end goal of the use case to be achieved. A secondary actor never initiates the use case. It is invoked by the system's use cases in order to obtain the required information or result. There may be many secondary actors for a given system.

Sequence Diagram
A sequence diagrams is a UML diagram that depicts interactions among various application components or participants over time, including but not limited to system objects, actors, and other systems or services, in order to accomplish a task.

SIPOC Diagram
The SIPOC diagram is a tool that is used to outline the scope of a process improvement initiative (often as part of a Six Sigma improvement project). The tool captures all of the relevant elements of the process under consideration. The diagram's name is an acronym for the elements that need to be identified and documented. (S) – Suppliers: Who supplies the inputs to the process under consideration, (I) – Inputs: What are the inputs to the process, (P) – Process: What are the steps of the process that is being improved upon, O – Outputs: What are the outputs of the process, C – Customers: Who are the customers or beneficiaries of the outputs of the process.

Six Sigma
Six Sigma is a process improvement methodology. It is structured into 5 phases which can be iterated to continually improve key processes and deliver greater efficiencies and success within an organization. These 5 phases are Define, Measure, Analyze, Improve, and Control.

Stakeholder Analysis
Stakeholder Analysis is the process of identifying project stakeholders, how their needs may impact the project, and the contributions that the stakeholders will make to the requirements elicitation process.

Structured English
See Pseudocode.

SWOT Analysis
SWOT Analysis is a strategic planning technique used to assess the internal and external environment in which a company operates and competes. Internal environmental factors are classified into strengths and weaknesses, while external environmental factors are classified into opportunities and threats.

T

U

UI Design Pattern
See User Inteface Design Pattern.

Use Case Diagram
A use case diagram is a UM L diagram that provides a high-level graphical view of the functionality (use cases) supported by the system and shows which roles (actors) can invoke each use case. This high-level view of the system provides a context for the readers of the more detailed use case specifications.

Use Case Specification
The use case specification provides the details of the functionality that the system will support and describes how the actors will use the system in order to obtain a specific result of value.

User Interface Design Patterns
User Interface Design Patterns (also commonly referred to as Interaction Design Patterns) document and convey robust UI design solutions, that have proven to be successful over time, to common usability requirements. Properly applying UI Design Patterns ensures the UI designer that the application or website will be intuitive and its features and functionality robust.

User Story
A user story (typically used by Agile methodolgies) is a high-level requirement containing just enough information to help the team produce a reasonable sizing for the requirement. The user story is generally one to two sentences in the everyday language of the user.

V

View
A view organizes diagrams into logical groups to describe a particular aspect of the system. It is the abstraction of the system organized is such a way as to give a perspective of a related set of concerns.

W

Work Breakdown Structure
The Work Breakdown Structure (WBS) documents the subdivision of tasks and effort required to complete an objective or project. It is most often depicted as a tree structure where high level tasks break down into lower level tasks. Low level tasks are typically grouped in various logical ways such as by system, subsystems, project phase, or a combination of these.

X

XML
XML stands for EXtensible Markup Language. XML was designed to transport and store data. It is a self descriptive markup language. This means that the tags used to describe the content of the XML file are not predefined, but instead the author defines his own tags and document structure.

Y

Z