Requirements Analysis

This topic covers the Requirements Analysis Technical Process, its key inputs and outputs, and illustrates how it can be applied.

Get the printer friendly version of the topic here.

Contents of Topic

Approximate Length: 2 hours

Topic Description:

Requirements Analysis is a Systems Engineering Technical Process that is a key bridge between the Stakeholder Requirements Definition and Architecture Design Processes.

The Requirements Analysis Process obtains sets of logical solutions (functional architectures) so as to improve understanding of the system Technical Requirements and the relationships among them. Performance parameters and constraints are allocated to these logical solutions, candidate functional architectures are developed, and Derived Technical Requirements that later become the basis for the system design are defined.

This topic discusses the Requirements Analysis Process, illustrates some of the ways it can be performed and how it can be used.

Select NEXT to continue.

D

D-Link Text:

Long Description:

The animation begins with a puzzle, which includes a piece labeled 'Requirements Analysis.' This ends with a display of the DAU logo.

Close window to continue.

Popup Text:

Stakeholder Requirements Definition
One of the Systems Engineering Technical Processes, Stakeholder Requirements Definition is used to determine Stakeholder Requirements (customer and other interested party requirements) and use them as the basis to develop well written Technical Requirements for a given system model.

Architecture Design
One of the Systems Engineering Technical Processes, Architecture Design is used to transform the outputs of the Requirements Analysis Process into a set of design solutions and a physical architecture representing a feasible system solution that is described by specifications and other design configuration descriptions.

Requirements Analysis Topic Objectives

Listed below are the objectives for this topic:

- Describe the purpose, inputs and outputs, and activities of the Requirements Analysis Process
- Outline the relationship of the Requirements Analysis Process to the Stakeholders Requirements Definition and Architecture Design Technical Processes
- Distinguish between Derived Requirements and Derived Technical Requirements
- Describe why bi-directional traceability is important
- Describe tools and techniques that can be used to assist in the Requirements Analysis Process
- Explain the role of Quality Function Deployment

D

Select NEXT to continue.

D-Link Text:

Long Description:

Chart that reads: Objectives: Describe the purpose, inputs and outputs, and activities of Requirements Analysis, Outline the relationship of Requirements Analysis to the Stakeholder Requirements Definition and Architecture Design Technical Processes, Distinguish between Derived Requirements and Derived Technical Requirements, Describe why bi-directional traceability is important, Describe tools and techniques that can be used to assist in the Requirements Analysis Process, and Explain the role of Quality Function Deployment.

Close window to continue.

Requirements Analysis Process

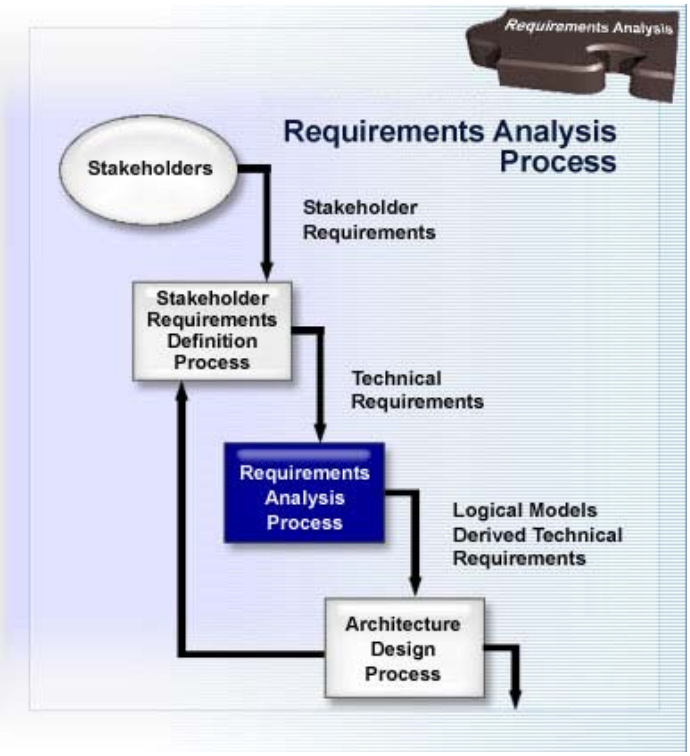
The outcomes of the Stakeholder Requirements Definition Process include a baselined set of Technical Requirements for a system.

These Technical Requirements are necessary but not sufficient for generating the ultimate design solution. Requirements Analysis enables a more detailed definition and understanding of the problem to be solved.

The Requirements Analysis Process transforms the set of baselined Technical Requirements into:

1. Logical Models making up the system's functional architecture that define and help in understanding the relationships among the functions, behaviors, data flows, attributes, services and states and modes of the Technical Requirements

2. Derived Technical Requirements that are formed from Logical Model(s) analysis and then allocated to system model elements



Select NEXT to continue.

Popup Text:

Technical Requirements
Technical Requirements establish the basis for system development. They are derived from analysis of Stakeholder Requirements and expressed in confirmed and quantitative 'shall' statements. They are used for verifying the physical design solution and are an input to the Requirements Analysis Process.

Logical Models
A logical model is one of the outputs of the Requirements Analysis Process. A logical model is a representation of the relationship of Technical Requirement attributes required to be satisfied by the later physical solution. Each of these attributes can be represented by a model. Collectively these models represent the functional architecture of a system.

For example, the attributes and their associated models (shown in parentheses) can include: functional (functional flow or logic diagrams), behavioral (response diagrams), temporal (time-based sequencing), product states and modes (transition diagrams), data (data flows), controls (control flows), failure modes (functional failure modes and effects) and information (information flows).

Derived Technical Requirements
Derived Technical Requirements are those that result from the analysis and allocation of Technical Requirements to logical functional architectures that are developed as part of the Requirements Analysis process; or from the analysis of alternative solutions done later as part of the Architecture Design Process. Derived Technical Requirements become the basis for the solution-specified requirements for the system model.

Importance of the Functional Architecture

Collectively the various logical models developed in performing Requirements Analysis are the system's functional architecture. Developed using a variety of functional analysis techniques, some of the essential benefits provided by a functional architecture are listed here.

Additionally, definition of the system functional architecture enables:

- Better understanding of system key interfaces
- Improved insight into system integration
- Support for development of verification and validation criteria for system elements in relation to Stakeholder Requirements, including traceability
- Performance of "what-if" scenarios to explore stressing Conditions to evaluate possible system risks

Select NEXT to continue.

D

D-Link Text:

Long Description:

A chart with the following key points:

A Functional Architecture helps provide:

- A measure of the system's ability to fulfill its functional objectives
- The system's ability to operate within given resource constraints
- Insight to life cycle economic and other system costs
- An ability to optimize trade-offs by understanding the implications and side effects---both positive and negative---associated with various architectural options

Close window to continue.

Types of Derived Requirements

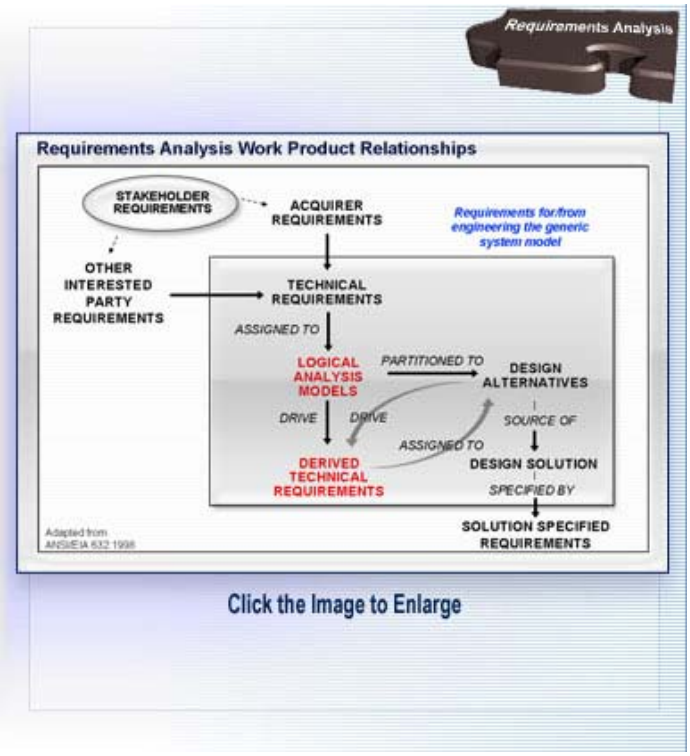
Some requirements are not explicitly stated in a higher-level set of requirements yet must be satisfied in order to provide a complete system solution. Types of these requirements include:

- **Derived Requirement:** One that is not explicitly stated in the set of Stakeholder Requirements yet is required to satisfy one or more of them. Many of these are addressed as part of the Stakeholder Requirements Definition Process.

- **Derived Technical Requirement:** One that results from allocation of Technical Requirements to logical models or from analysis of alternative solutions. A Derived Technical Requirement is a 'design-to' requirement for the system.

The graphic highlights how the requirements that are developed using the three System Design processes relate to each other. Hint!

Select NEXT to continue.



D

D-Link Text:

Long Description:

Flowchart titled 'Requirements Analysis Work Product Relationships'. Stakeholder Requirements can flow to either Acquirer Requirements or Other Interested Party Requirements. Both of these flow to Technical Requirements, then are assigned to Logical Analysis Models, which can then flow either to Derived Technical Requirements or Design Alternatives. Design Alternatives are a source of Design Solution specified by Solution-Specified Requirements.

Close window to continue.

Popup Text:

these
Derived Requirement Example:

Stakeholder Requirement: A missile must have a fly-out distance of 2 kms and hit a target having a Radar Cross Section the size of a tennis ball.

Derived Requirement: The missile shall be aimed within 2 degrees of the target so that the warhead terminal seeker can lock on and perform the terminal intercept.

Hint!
Software-intensive systems are particularly rich in Derived Technical Requirements. Keeping track of them is especially important and failure to do so inevitability has led to significant software development problems and unplanned code growth later on.

Use of an effective Requirements Management Technical Management Process + the use of Technical Reviews such as the Software Specification Review (SSR) to keep a handle on the growth of these types of requirements is key in keeping this critical part of system development under control!

Knowledge Review

Select the two correct responses.

What types of requirements are not explicitly stated in a higher-level set of requirements but still must be satisfied to provide a complete solution?

- A. Other Interested Party Requirements that are derived from the Design Solution
- B. Derived Requirements needed to satisfy one or more Stakeholder Requirements
- C. Acquirer Requirements that are derived from Stakeholder Requirements
- D. Derived Technical Requirements resulting from analysis involved as part of the allocation of Technical Requirements to logical models

Submit

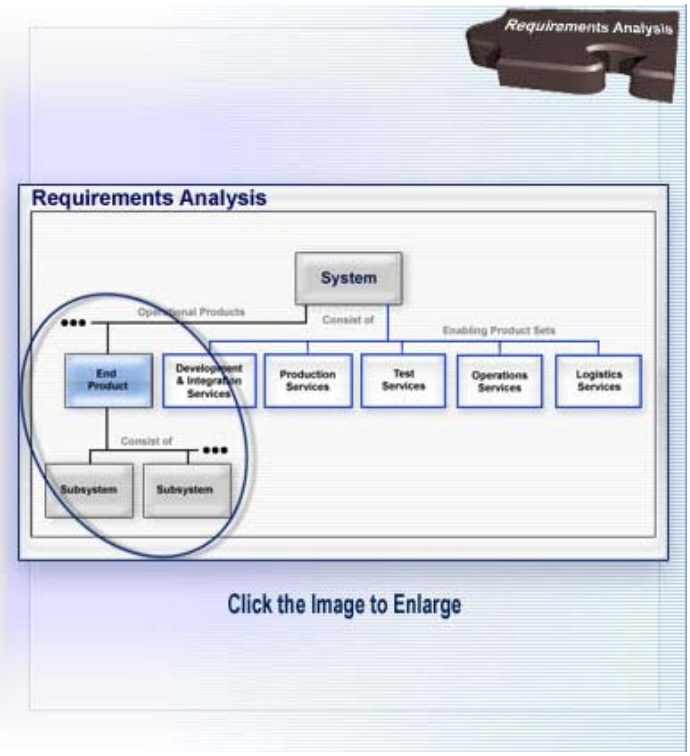
Purpose of Requirements Analysis

Purpose: To (1) obtain sets of logical solutions [functional architectures] derived from defined Technical Requirements, and (2) to better understand the relationships among them. Once these logical solution sets are formed, allocated performance parameters and constraints are set, and the set of Derived Technical Requirements for the system can be initially defined.

Outcomes: For each end product of a system model:

1. A set of logical models showing relationships (e.g., behavioral, functional, temporal, data flows, etc.) among them
2. A set of 'design-to' Derived Technical Requirements

Select NEXT to continue.



D

D-Link Text:

Long Description:

Graphic titled 'Requirementsl Analysis'. Flowchart begins with System and branches off to Operational Products and Enabling Product Sets. Operational Products branches off to End Product. Enabling Product Sets branches off to Development & Integration Services, Production Services, Test Services, Operation Services and Logistics Services. End Product branches off to two blocks labeled 'Subsystem.'

Close window to continue.

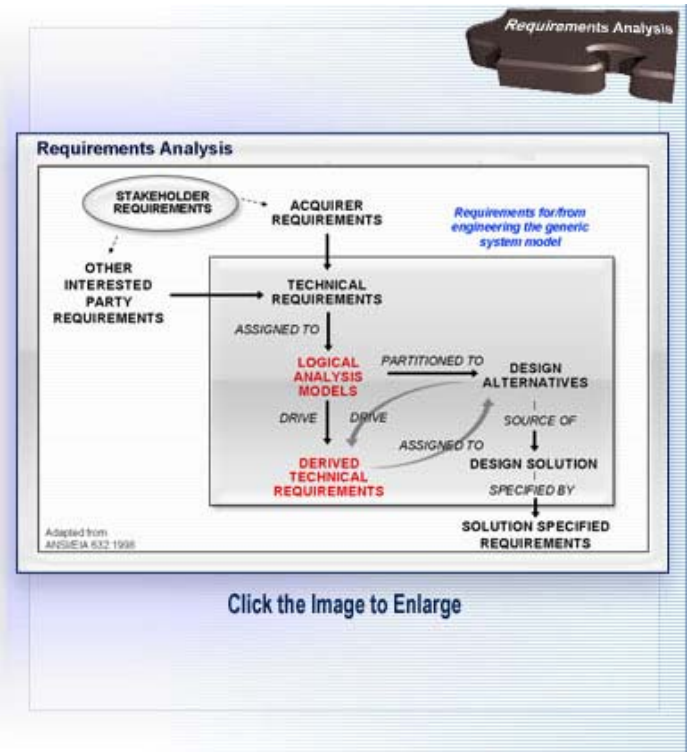
Requirements Analysis Process Focus

The Requirements Analysis Process encompasses the formation of functional architectures, the allocation of Technical Requirements to them and uses the results of the allocation analysis process to develop Derived Technical Requirements.

The approach resulting from the Requirements Analysis Process:

- Partitions (allocates) a system into self-contained, cohesive, logical groupings of interchangeable and adaptable elements to enable ease of change, achieve technology transparency and mitigate the risk of obsolescence
- Uses rigorous and disciplined definitions of interfaces and, where appropriate, defines the Key Interfaces within a system using widely supported, open system standards

Select NEXT to continue.



D

D-Link Text:

Long Description:

Flowchart titled 'Requirements Analysis Work Product Relationships'. Stakeholder Requirements can flow to either Acquirer Requirements or Other Interested Party Requirements. Both of these flow to Technical Requirements, then are assigned to Logical Analysis Models, which can then flow either to Derived Technical Requirements or Design Alternatives. Design Alternatives are a source of Design Solution specified by Solution-Specified Requirements.

Close window to continue.

Popup Text:

Key Interfaces
A Key Interface is a common boundary shared between system modules that provides access to critical data, information, materiel or services; and/or is of high interest due to rapid technological change, a high rate of failure or costliness of connected modules.

open system
An Open System is one that implements specifications maintained by an open, public consensus process for interfaces, services and support formats. This is done to enable properly engineered components to be utilized across a wide range of systems with minimal change, to interoperate with other components on local and remote systems, and to interact with users in a manner that facilitates portability.

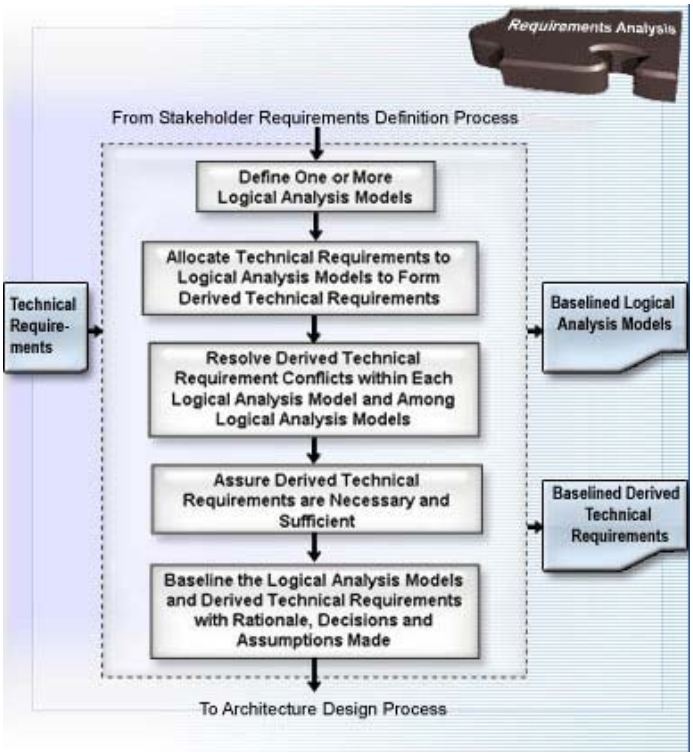
Requirements Analysis Process Inputs and Outputs

The key inputs and outputs of the Requirements Analysis Process are illustrated here and include:

1. **Key Inputs**: Technical Requirements from the Stakeholder Requirements Definition Process
2. **Key Outputs**: Logical Analysis Models and Derived Technical Requirements that are inputs to the Architecture Design Process

In addition, the various work products produced by process activities are recorded in the technical data management database.

Select NEXT to continue.



D

D-Link Text:

Long Description:

A flowchart beginning with the Stakeholder Requirements Definition Process. The first step is to define one or more logical analysis models, then allocate technical requirements to logical analysis models to form derived technical requirements, then resolve derived technical requirement conflicts within each logical analysis model and among logical analysis models. Then assure derived technical requirements are necessary and sufficient, then baseline the logical analysis models and derived technical requirements with rationale, decisions and assumptions made. The flow can then go to Architecture Design process or to baselined derived technical requirements and baselined logical analysis models.

Close window to continue.

Popup Text:

work products
Examples of the work products of logical analysis include the following types of logical analysis models:

- Functional flow block diagrams
- Logic flow diagrams
- Time lines
- Context diagrams
- State and mode transitions
- Structured analyses algorithms
- Data flows
- Services and attributes

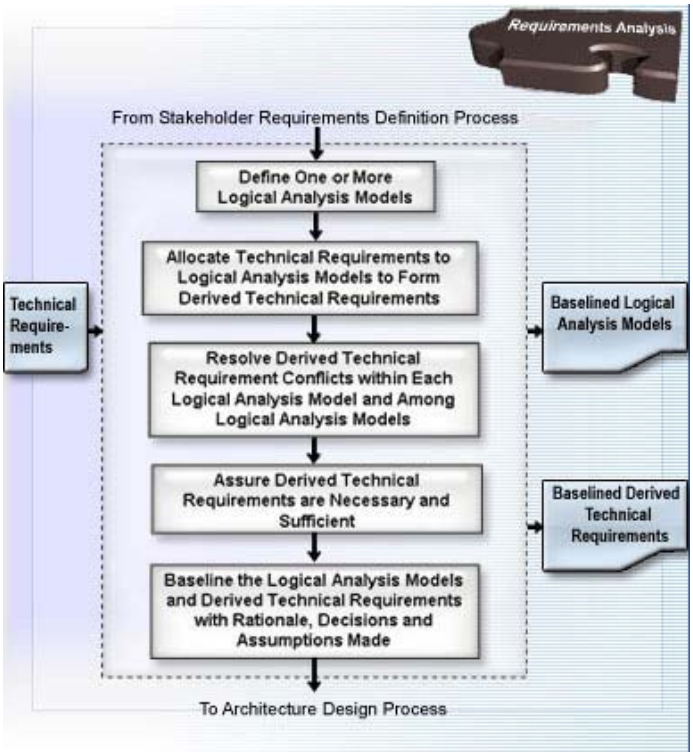
- Behavioral diagrams and responses
- Requirements allocation sheets

Requirements Analysis Key Activities

The key activities that make up the Requirements Analysis Process are:

1. **Define**: One or more logical analysis models
2. **Allocate**: Technical Requirements to logical analysis models to form Derived Technical Requirements
3. **Resolve**: Derived Technical Requirement conflicts with each logical analysis model and among logical analysis models
4. **Assure**: Derived Technical Requirements are necessary and sufficient
5. **Baseline**: The logical analysis models and Derived Technical Requirements with rationale, decisions and assumptions

Select NEXT to continue.



D

D-Link Text:

Long Description:

A flowchart beginning with the Stakeholder Requirements Definition Process. The first step is to define one or more logical analysis models, then allocate technical requirements to logical analysis models to form derived technical requirements, then resolve derived technical requirement conflicts within each logical analysis model and among logical analysis models. Then assure derived technical requirements are necessary and sufficient, then baseline the logical analysis models and derived technical requirements with rationale, decisions and assumptions made. The flow can then go to Architecture Design process or to baselined derived technical requirements and baselined logical analysis models.

Close window to continue.

Knowledge Review

Please select a correct answer.

All of the following activities are performed during the Requirements Analysis Process **except** _____.

- A. Define one or more logical analysis models.
- B. Allocate Technical Requirements to logical analysis models.
- C. Translate user needs into system Technical Requirements.
- D. Baseline the logical analysis models and Derived Technical Requirements.

Submit

Requirements Analysis Models

Activity: Define Logical Analysis Models

Tasks:

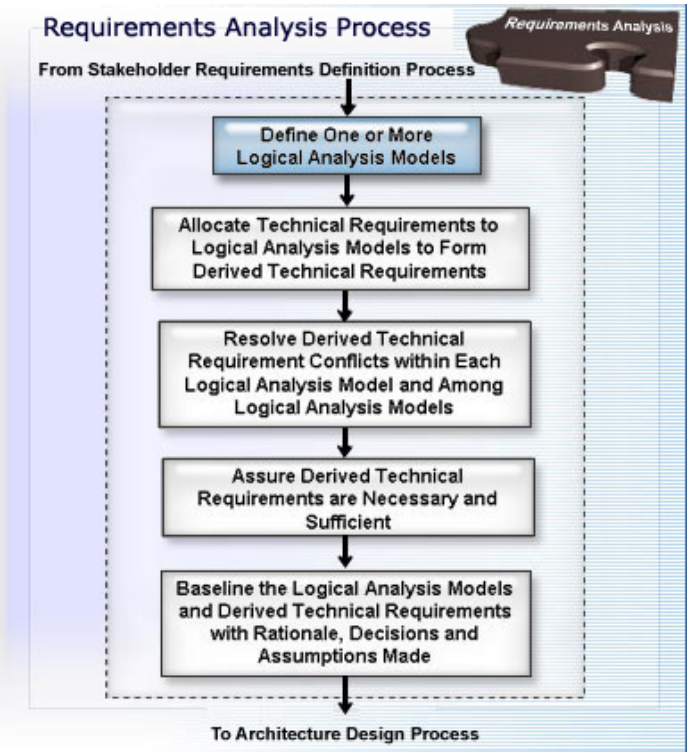
1. **Logical Models**: Establish logical models for the defined Technical Requirements by conducting trade studies against cost, schedule, performance and risk using the Decision Analysis Process.

2. **Functional Analysis**: Perform functional analysis to understand the subfunctions and their logical flows.

3. **Identify & Define**: Identify interfaces, states and modes, timelines and data control flows.

4. **Behavior Analysis**: Analyze behaviors, operator tasks and functional failure modes and effects, as appropriate.

Select NEXT to continue.



D

D-Link Text:

Long Description:

Flow begins with the Stakeholder Requirements Definition Process. The first step is to define one or more logical analysis models (highlighted), then allocate technical requirements to logical analysis models to form derived technical requirements, then resolve derived technical requirement conflicts within each logical analysis model and among logical analysis models. Then assure derived technical requirements are necessary and sufficient, then baseline the logical analysis models and derived technical requirements with rationale, decisions and assumptions made. The flow then goes to Architecture Design process.

Close window to continue.

Popup Text:

Decision Analysis Process
One of the Systems Engineering Technical Management Processes, Decision Analysis Process activities provide the basis for evaluating and selecting alternatives when decisions need to be made.

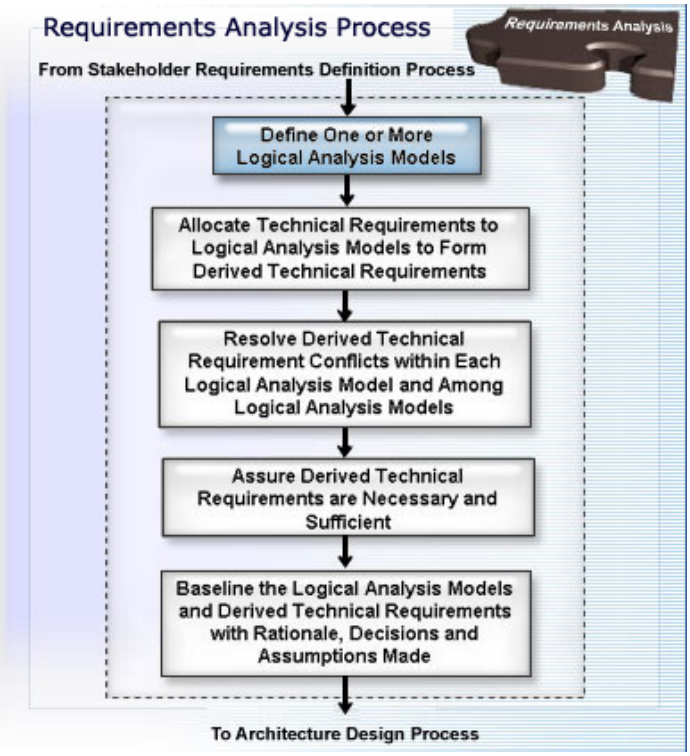
Logical Analysis Models, Cont.

Expected Outcomes:

Logical Analysis Models. Examples include:

- Functional flow, timelines, behaviors, data and control flows, states and modes, functional failure modes and effects
- Model data and functions with algorithms used to decompose functions while explicitly showing the data needed for each function
- Data structures with their functions and processing flows

Select NEXT to continue.



D

D-Link Text:

Long Description:

Flow begins with the Stakeholder Requirements Definition Process. The first step is to define one or more logical analysis models (highlighted), then allocate technical requirements to logical analysis models to form derived technical requirements, then resolve derived technical requirement conflicts within each logical analysis model and among logical analysis models. Then assure derived technical requirements are necessary and sufficient, then baseline the logical analysis models and derived technical requirements with rationale, decisions and assumptions made. The flow then goes to Architecture Design process.

Close window to continue.

Popup Text:

Examples
The Department of Defense Architecture Framework (DoDAF) is one way of documenting some of these models. The DoDAF consists of Operational Views (OVs), which identify warfighter information needs; System Views (SVs), which overlay capabilities on requirements; and Technical Views (TVs), which identify the rules, standards and conventions used to interface system products. These views are formally described by a series of over 20 documents. Some DoDAF products are both outputs of the JCIDS process as well as required submissions by the acquirer at milestone reviews.

In many cases more details and granularity are needed to adequately perform Requirements Analysis than provided by the DoDAF product sets. The format or form selected by the engineering team is that which best defines the functional, behavior, or data flow or data structure, as appropriate for the problem at hand. It should be one that will allow best assignment to potential end products, manual operations, or enabling

products as well as for generating alternatives.

Logical Analysis Models - Considerations

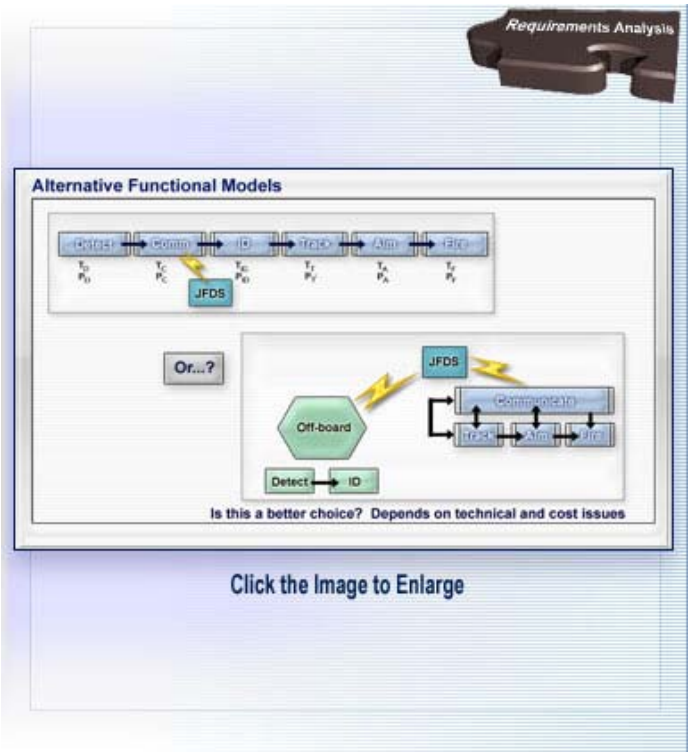
One or more of the analysis tools and techniques described on the following screens can be used to create models to help understand the relationships among functions and other attributes embedded in Technical Requirements.

Trade studies should be conducted to determine the most useful models from a cost/performance perspective.

One key consideration for such trade studies is the best way to partition the logical models representing the system. A common rule is to 'maximize cohesion' and 'minimize coupling.' This means:

- Cohesion: Encapsulate only similar functions
- Coupling: Minimize dependent interfaces

Such considerations in forming Logical Analysis Models are important enablers for later successful use of an Open System design approach.



D

Select NEXT to continue.

D-Link Text:

Long Description:

Graphic titled 'Functional Architecture.' There are two graphic boxes. The first box is a flowchart that begins with Detect, then, Comm, ID, Track, Aim and Fire. The second box has Off-board, Detect, ID. There is also Communicate, Track, Aim, Fire. At the bottom a sentence reads: Is this a better choice? Depends on technical and cost issues.

Close window to continue.

Popup Text:

Open System
An Open System is one that that implements specifications maintained by an open, public consensus process for interfaces, services and support formats. This enables properly engineered components to be utilized across a wide range of systems with minimal change, to interoperate with other components on local and remote systems, and to interact with users in a manner that facilitates portability. Done properly, use of Open Systems reduces development costs, allows for graceful technology insertion over the life of the system and reduces life cycle costs.

What Is Functional Analysis?

Functional analysis is the systemic identification and description of all functions--in terms of inputs, outputs and interfaces--that must be done to fulfill system requirements. Functional analysis includes:

- Translating top-level requirements into functions needed to accomplish them
- Decomposing and allocating these functions to lower levels of the system structure
- Identifying and describing functional and subsystem interfaces

Functions are arranged in logical sequences so operational system usage can be traced end-to-end. The process continues until all desired system levels have been analyzed, defined and baselined. The result is the system function architecture.

Select NEXT to continue.

Functional Analysis

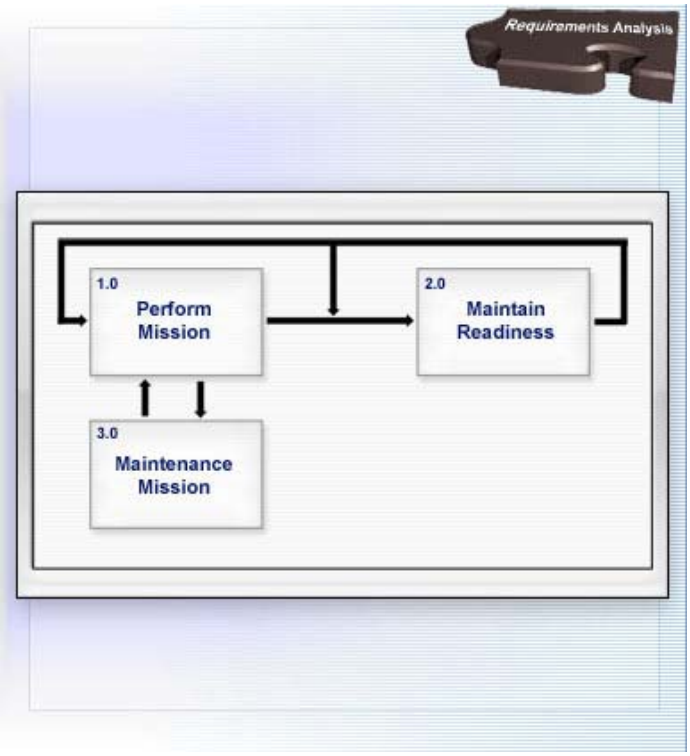
When using functional analysis, the ultimate output is a set of system-level functions to be performed by the system's End Products, sequenced by time.

This functional picture of the system details the complete set of functions to be performed, along with their relationships.

It represents the functional architecture of the system.

Aside from the creative minds of the system designers and their domain knowledge, a variety of tools and techniques can be used to assist in functional analysis. Some of them are described in the following set of screens.

Select NEXT to continue.



Popup Text:

End Products
An end product is defined by a customer need and performs the operational functions required by a customer.

functional architecture
This is an arrangement of functions and their subfunctions and interfaces (internal and external) that defines the execution sequencing, conditions for control or data flow, and the performance requirements to satisfy the requirements baseline (IEEE Std 1220).

Functional Flow Block Diagram

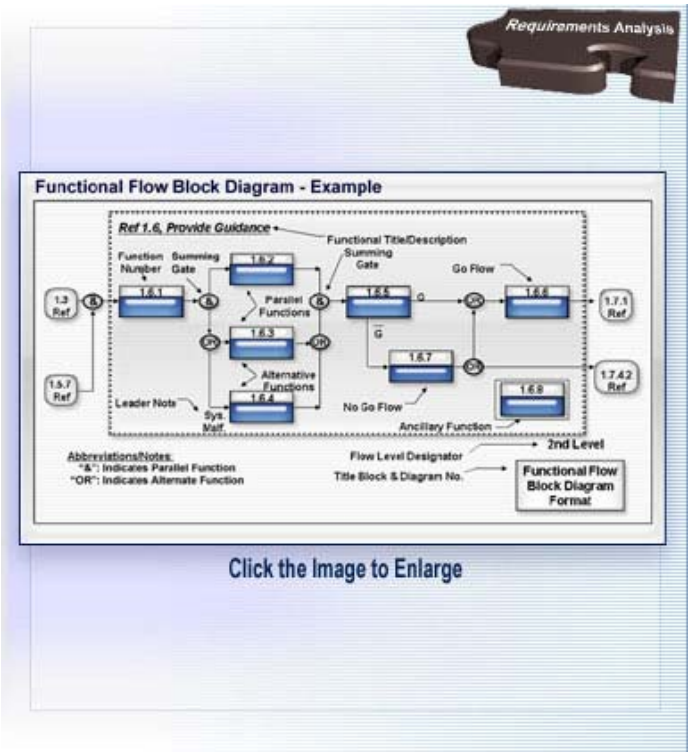
A basic technique used in functional analysis is the Functional Flow Block Diagram (FFBD).

In a FFBD, individual function statements are used that represent the discrete actions (which are described initially by subject-verb-object statements) that an end product of a system model must perform.

These function statements are then ultimately reduced to a verb that best expresses the action needed for functional analysis and FFBD work.

The functional flow shows which functions must either precede or follow others as well as which ones can be accomplished in parallel.

Select NEXT to continue.



D

D-Link Text:

Long Description:

A graphic titled ' Functional Flow Block Diagram-Example'. Function Number 1.6.1 flows to 1.6.2, 1.6.3 and 1.6.4. They flow to 1.6.5, 1.6.6 or 1.6.7. 1.6.8 is an ancillary function. 1.6.2 and 1.6.3 are parallel functions. 1.6.3 and 1.6.4 are alternative functions.

Close window to continue.

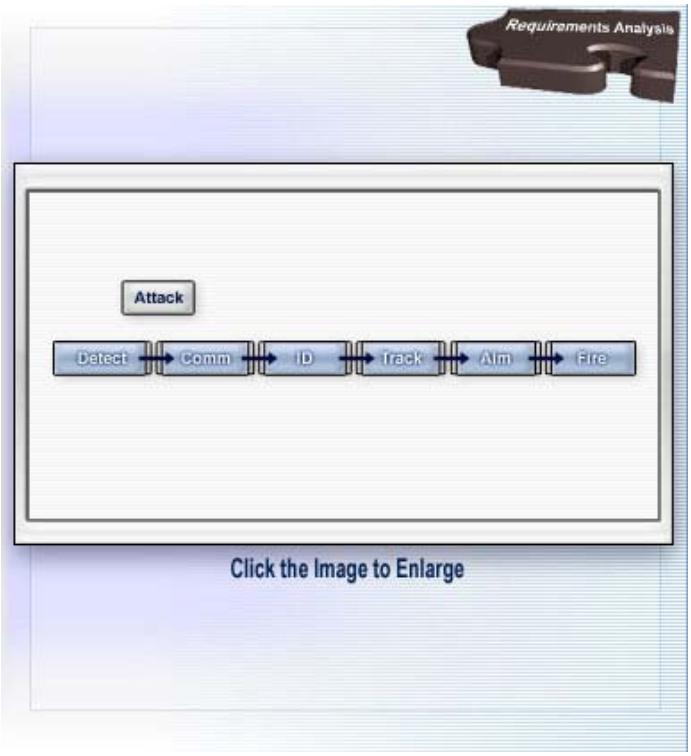
Functional Decomposition

In an FFBD, individual functions are used that represent discrete actions (described by verbs) that a system must perform to carry out its mission. In the case illustrated here, the 'Attack' function is broken down into its constituent subfunctions, consistent with the system's overall Concept of Operation (CONOPS).

Even though shown as a sequence of linear sub-functions in the graphic, in arranging them logically, some of them could conceivably be operated in parallel (e.g., the Comm and ID subfunctions). The purpose is to determine the 'best' ordering of sub-functions to accomplish the system function of "Attack."

The CONOPS or other customer requirements would not typically determine such an optimal ordering. This determination is part of the Requirements Analysis Process. Additionally, the domain expertise of IPT members is especially critical for effective and credible functional decomposition.

Select NEXT to continue.



Popup Text:

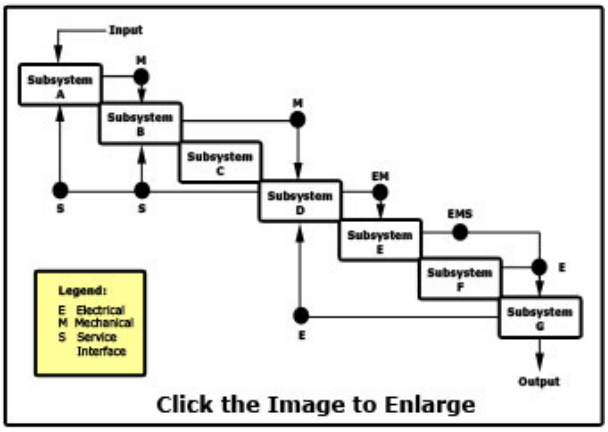
Concept of Operation (CONOPS)
A Concept of Operations (CONOPS) is based on the envisioned operational use of the End Product. It involves documenting the steps involved at the beginning of use and thinking it through to the end. A CONOPS is one of the key outputs of the Stakeholder Requirements Definition Process.

N2 Chart

An N-squared (or N2) chart can be used to help define functional interfaces. As illustrated here, in an N2 Chart, system components or functions are placed on the diagonal; the remainder of the linkages represent various interface inputs and outputs. The N2 Chart complements the FFBD by highlighting the data flows as inputs and outputs of system functions.

The N2 Chart is taken down into successively lower levels of the system structure, ultimately down to component functional levels.

The N2 Chart also pinpoints areas where conflicts could arise in functional interfaces and highlights input and output dependency assumptions and requirements.



D

Select NEXT to continue.

D-Link Text:

Long Description:

The N2 Chart illustrates Subsystems A-G arranged diagonally down the page. Interfaces among them are shown, using lines between the subsystems that need to interface. The type of interface is also noted: Electrical, Mechanical and/or Service.

Close window to continue.

Knowledge Review

Please select a correct answer.

A technique that analyzes the sequenced functional flows that a system must perform to carry out its mission is called _____.

- A. Functional Flow Block Diagram
- B. Timeline Analysis
- C. Object-Oriented Analysis
- D. Requirements Allocation Sheet

Submit

Knowledge Review

Please select a correct answer.

Illustrated here is a functional analysis of a car pulling out of its garage. Which is the most appropriate function for the empty block?

- A. Insert key into ignition.
- B. Roll down window.
- C. Turn on radio.
- D. Shift transmission into gear.

Submit

D

D-Link Text:

Long Description

A flowchart titled 'Review'. At the top there is a car with a text box that reads 'Pull out of garage.' The flow boxes read: Enter Car, then Start Engine, blank box, Release Accelerator, Depress Brake.

Close window to continue.

Knowledge Review

Please select a correct answer.

The functional analysis is continued to the next level for 'Start Engine'. For the first empty block, which of the following is the appropriate function?

- A. Insert key into ignition.
- B. Shift transmission into gear.
- C. Turn on radio.
- D. Open garage door.

Submit

D

D-Link Text:

Long Description

A flowchart titled 'Review'. At the top there is a car with a text box that reads 'Pull out of garage.' The flow boxes read: Enter Car, then Start Engine, Shift Transmission into Gear, Press Accelerator, Release Accelerator, Depress Brake, blank box, Turn to Start, Release to Run when Engine Starts.

Close window to continue.

Timeline Analysis

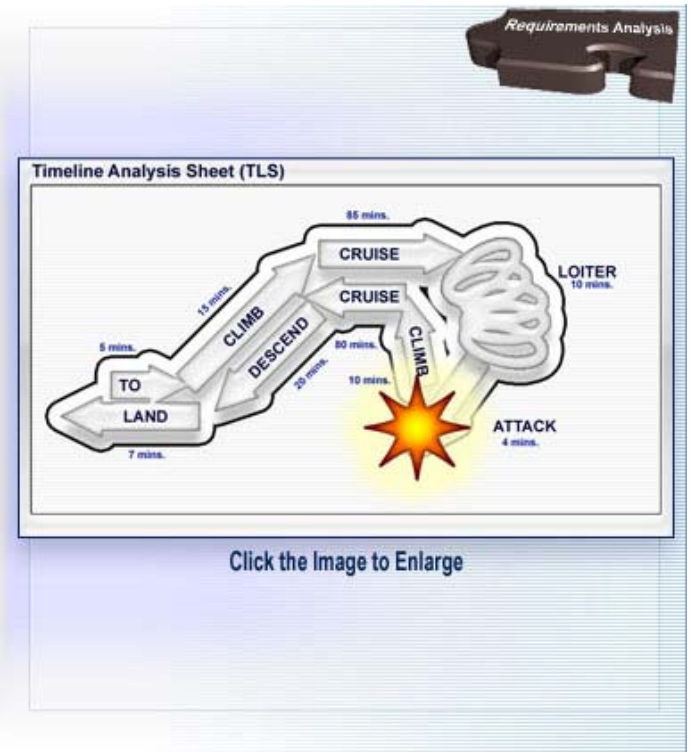
When using a functional approach, the ultimate output is a set of all functions that are to be performed by the system, sequenced by time.

A technique that can be used to aid in determining that sequencing is called Timeline Analysis.

Timeline Analysis defines the duration and sequence of various time-critical functions.

An example for a Remotely Piloted Vehicle (RPV), showing its critical functions from Take-Off (TO) to Landing, is illustrated here.

Select NEXT to continue.



D

D-Link Text:

Long Description:

Graphic titled 'Timeline Analysis Sheet (TLS)'. Roadmap that begins with TO (5 minutes), Climb (15 minutes), Cruise (85 minutes), Loiter (10 minutes), Attack (4 minutes), Climb (10 minutes), Cruise (80 minutes), Descend (20 minutes), Land (7 minutes).

Close window to continue.

Sample Timeline

The diagram shows a sample timeline for a man from the time he gets up until he leaves for work.

If the requirement was that the person leaves in 30 minutes or less from the time the alarm went off, this would be an acceptable logical solution.

Is this your morning timeline? Probably not!

This demonstrates that there are many different potential logical solutions to the same set of Technical Requirements. Many valid analyses can result from the same set of requirements.

Outputs of the Stakeholder Requirements Definition process such as the CONOPS and performance measures (e.g., MOEs, MOSs, COIs, CTPs, KPPs) play an essential role in 'bounding the solution space' as Requirements Analysis proceeds.

Select NEXT to continue.

Popup Text:

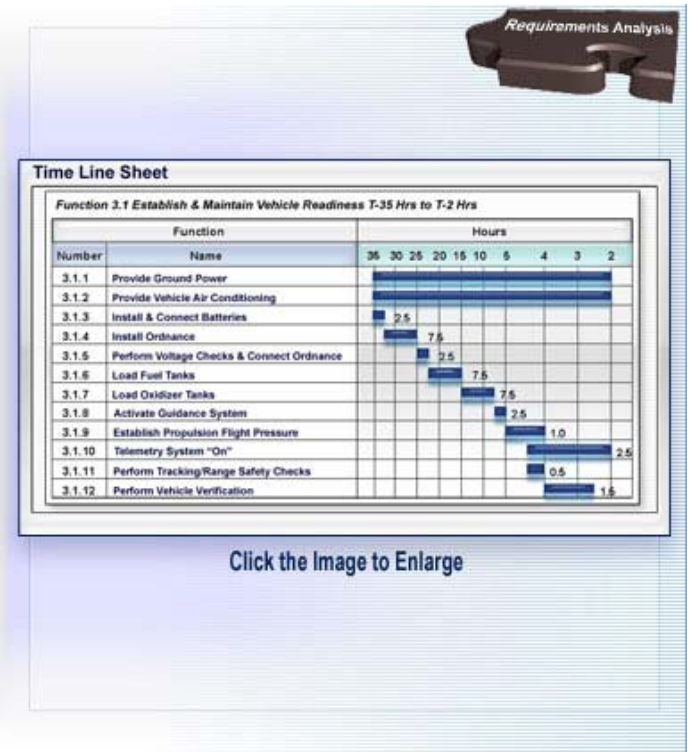
'bounding the solution space'
Another way of stating this, according to some authors, is eliminating dumb ideas as early as possible!

Time Line Sheet

The Time Line Sheet (TLS) adds detail to defining durations of various functions. It defines concurrency, overlapping and sequential relationships of functions and tasks.

A TLS identifies time-critical functions that directly affect system availability, operating time and maintenance downtime. It is used to identify specific time-related design requirements.

Select NEXT to continue.



D

D-Link Text:

Long Description:

Chart titled 'Time Line Sheet'. Subheading is 'Function 3.1 Establish & Maintain Vehicle Readiness T-35 Hrs to T-2 Hrs.' Chart is broken into Function and Hours. Number 3.1.1--Provide Ground Power, 2 hours. 3.1.2--Provide Vehicle Air Conditioning, 2 hours. 3.1.3--Install & Connect Batteries, 2.5 hours. 3.1.4--Install Ordnance, 7.5 hours. 3.1.5--Perform Voltage Checks & Connect Ordnance, 2.5 hours. 3.1.6--Load Fuel Tanks, 7.5 hours. 3.1.7--Load Oxidizer Tanks, 7.5 hours. 3.1.8--Activate Guidance System, 2.5 hours. 3.1.9--Establish Propulsion Flight Pressure, 1.0 hour. 3.1.10--Telemetry System "On," 2.5 hours. 3.1.11--Perform Tracking/Range Safety Checks, 0.5 hour. 3.1.12--Perform Vehicle Verification, 1.5 hours.

Close window to continue.

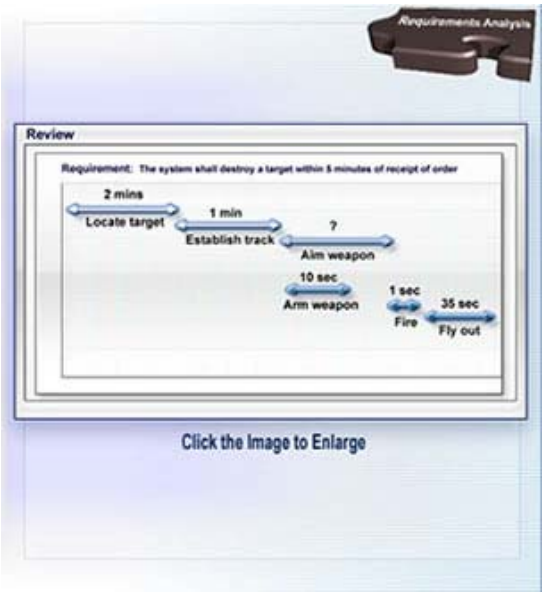
Knowledge Review

Please select a correct answer.

Mission Requirement: The system [an air-to-ground missile system fired from an unmanned aerial vehicle] shall destroy a target within 5 minutes of receipt of order.
Given this 'worst case' Timeline, what is the maximum time allowed for the "Aim Weapon" function?
Note: 'Fly Out' time is measured from missile launch to target impact at maximum range.

- A. 84 seconds
- B. 74 seconds
- C. 119 seconds
- D. 222 seconds

Submit



D

D-Link Text:

Long Description:

Graphic titled 'Review.' At the top of the graphic is Requirement: The system shall destroy a target within 5 minutes of receipt of order.' The graphic consists of six consecutive lines labeled as: Locate target, 2 minutes. Establish track, 1 minute. Aim weapon, ?. Arm weapon, 10 seconds. Fire, 1 second. Fly out, 35 seconds.

'Arm weapon' and 'Aim weapon' are parallel events.

Close window to continue.

Other Techniques

While functional analysis works well for hardware-oriented systems, different approaches and different tools may be used to analyze the development of software or systems that include both hardware and software. Such highly-integrated, software-intensive systems dominate development activity in the DoD today.

Even when both hardware and software are part of a system, functional analysis can still be used as the primary analysis technique up until the assignment of functions to the software. Then alternative approaches/techniques described here may come into play.

Over time, different tools have been developed to help analyze software-intensive systems. Some of them involving so-called Structured Analysis have their roots in classical functional analysis while others are derivatives of different ways of clustering software based on an object view of the end product.

Object-oriented tools had their genesis starting with the development in the 1980's of Object-Oriented Programming (OOP), a language-specific technique that when properly applied, can dramatically improve productivity, quality and levels of software reuse.

A variety of object-oriented and other techniques can be used to supplement functional analysis. Some of them are described in the following set of screens.

Select NEXT to continue.

Popup Text:

object
In object-oriented programming, an object is an individual unit which is used as the basic building block of programs. These objects act on each other. This is in contrast to a traditional view, in which a program may be seen as a collection of functions, or simply as a list of instructions to the computer. Each object is capable of receiving messages, processing data, and sending messages to other objects.

Object-Oriented Analysis (OOA)

A variety of techniques evolved as object-oriented approaches became better understood. Ultimately, it was realized that Object-Oriented Analysis (OOA) was a necessary precondition to optimize the entire object-oriented software development process.

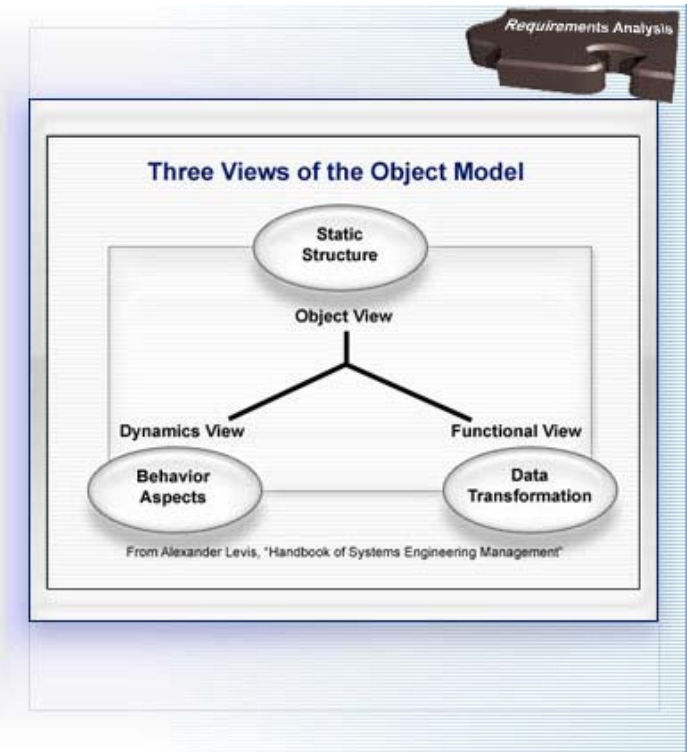
OOA creates three views of the object model:

- Object view - static structure
- Dynamics view - behavioral aspects
- Functional view - data transformation

Actual application of object-oriented techniques requires the use of sophisticated CASE tools and training in a specific approach or technique.

But the Requirements Analysis goal, whether for hardware or software or a combination of both, is the same: logical analysis models with performance adequately allocated, Derived Technical Requirements properly identified and a robust functional architecture.

Select NEXT to continue.



Popup Text:

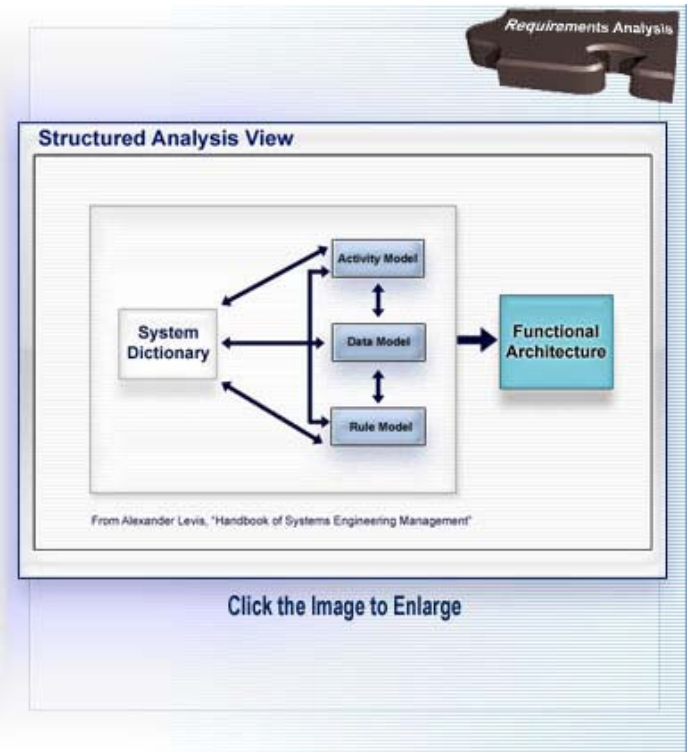
CASE
CASE (Computer-Aided Software Engineering) refers to the use of computers to aid in the software engineering process. CASE tools may include the application of software tools to software design, requirements tracing, code production, testing, document generation and other software engineering activities.

Structured Analysis

Structured Analysis uses three models that represent different views of the software product:

- **The Activity Model:** Made up of data flow diagrams, it is the primary view of the software product. It defines what is done, along with data flows, and provides the primary structure of the solution.
- **The Data Model:** It is made up of Entity-Relationship (ER) diagrams. It is a record of what is in the hardware/software set of products, or what is outside the set of products being monitored.
- **The Rule Model:** It is a dynamic view, made up of such items as state transition diagrams. It defines when things happen and the conditions under which they happen.

IDEF (ICAM Definition Language) diagrams are one way to model the flows of data within a system.



D

Select NEXT to continue.

D-Link Text:

Long Description:

Graphic titled 'Structured Analysis View'. In the graphic there is a box labeled System Dictionary, which branches off to three models: Activity Model, Data Model and Rule Model. All branch to Functional Architecture.

Close window to continue.

Popup Text:

Entity-Relationship (ER) diagrams
Entity-Relationship (ER) diagrams are graphical notations for representing data models. Such models can be used in the first stage of information-system design. They are used, for example, to describe information needs and/or the type of information that is to be used and processed.

ICAM
ICAM is short for Integrated Computer-Aided Manufacturing. ICAM was an initiative managed by the US Air Force out of Wright Patterson AFB, Materials Laboratory in the 1980's and was part of their Technology Modernization efforts, specifically the Computers in Manufacturing (CIM) initiative. The ICAM project office deemed it valuable to create a 'neutral' way of describing data structures so that a way could be found to process data independently of the way it was physically stored. The IDEF models were the result. IDEF originally was an acronym for 'I-CAM Definition Methods' but has evolved to current usage, which is synonymous with 'Integrated Definition.'

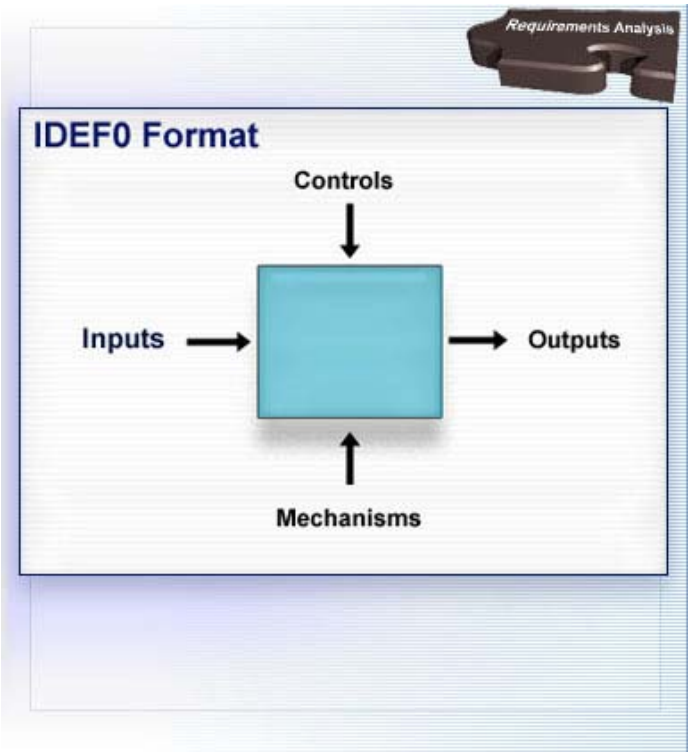
IDEF Diagrams

A number of IDEF products (e.g., IDEF0, IDEF1X...IDEF5) have been developed over the last twenty years. They can be used to represent data flows and model portions of a set of systems products (mostly computer/software products) from different perspectives.

Since the original IDEF standards were produced by government funds, the IDEF technique is in the public domain and has achieved fairly wide acceptance because of its non-proprietary legacy.

One standard method of conducting data flow analysis is the use of the IDEF0 modeling language. IDEF0 diagrams can be used to graphically depict structured representations of activities and functions.

Select NEXT to continue.



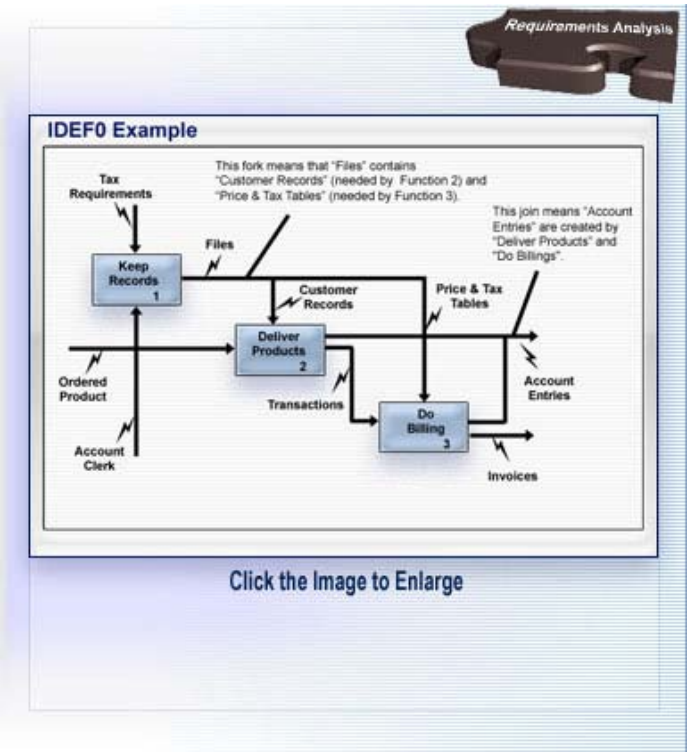
IDEF0 Example

Displayed is an example of an IDEF0 diagram representing a billing process.

Inputs, Outputs, Controls and Implementing Mechanisms are all shown on a single, comprehensive diagram.

This makes it easier to analyze the various relationships among the functions that a computer-based information or business system must perform.

Select NEXT to continue.



D

D-Link Text:

Long Description:

A chart titled 'IDEFO Example'. Flow begins with 'Tax Requirements,' which flows to 'Deliver Products' and then 'Do Billing.' There is a fork between 'Keep Records' and 'Deliver Products.' This fork means that 'Files' contains 'Customer Records' and 'Price & Tax Tables.' There is a join after 'Price & Tax Tables. This join means 'Account Entries' are created by 'Deliver Products' and 'Do Billing.'

Close window to continue.

UML and SysML

The Unified Modeling Language (UML) is a general-purpose modeling language that includes a graphical notation used to create abstract models of a system, referred to as UML models. These models were originally designed to specify, visualize, construct and document software-intensive systems. A variety of UML diagrams can be used to model different systems views:

- **Functional Requirements View:** Presents the functional requirements of the system from the user's point of view. It includes use case diagrams.
- **Static Structural View:** Presents the system static structure using objects, attributes, operations and relationships
- **Dynamic Behavior View:** Presents dynamic behavior by showing collaborations among objects and changes to the internal states of objects

Although initially developed to support object-oriented programming, UML is not restricted just to modeling software. UML also has been used for business process modeling as well as for Systems Engineering via an emerging UML extension called 'SysML'.

The Systems Modeling Language (SysML) supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems. SysML is an extension of a subset of the Unified Modeling Language (UML).

Both UML and SysML are evolving tool sets, defined in a variety of open source industry specifications. More information about UML and SysML is available in the References section.

Select NEXT to continue.

Popup Text:

use case diagrams
Use Case Diagrams present a graphical overview of system functionality in terms of 'actors' and their 'goals—represented as use cases—and any dependencies between them.

Behavioral Analysis

Once various logical models are created and partitioned appropriately, a final technique can be used to help determine the optimal functional architecture. This technique is called 'behavioral analysis.'

Behavioral analysis involves the simulation or stimulation of functional architectures utilizing operational scenarios to expose the model to stressful situations that reflect anticipated usage and environments.

This analysis helps to ensure development of a robust design solution that will meet the user's operational needs.

Select NEXT to continue.

Requirements Allocation Sheet

A Requirements Allocation Sheet (RAS), or its equivalent, documents the connection among allocated functions, allocated performance and the system.

A Requirements Allocation Sheet also provides traceability between the Requirements Analysis and Architecture Design and shows any disconnects.

While shown here as a paper product for illustrative purposes, during Requirements Analysis, literally thousands of requirements are being tracked and analyzed.

Typically, a set of automated tools that automate various Systems Engineering processes is used by the developer for this process.

Select NEXT to continue.

D

D-Link Text:

Long Description:

Chart titled 'Requirements Allocation Sheet (RAS)'. For 1.6.8 Provide guidance compartment cooling--the temperature in the guidance compartment must be maintained at the initial calibration temperature of +0.2 degrees Fahrenheit. The initial calibration temperature of the compartment will be between 66.5 and 68.5 degrees Fahrenheit. There are facility requirements. For 1.6.8.1 Provide chilled coolant (primary)--a storage capacity for 65 gallons of chilled liquid coolant (deionized water) is required. The temperature of the stored coolant must be monitored continuously. The stored coolant must be maintained within a temperature range of 40-50 degrees Fahrenheit for an indefinite period of time. The coolant supplies must be free of obstructive particles 0.5 micron at all times. The nomenclature is tank and the CI or item number is 26-2764.

Close window to continue.

Knowledge Review

Please select a correct answer.

A technique that involves the simulation or stimulation of functional architecture models utilizing operational scenarios reflecting anticipated stressful usage is _____.

- A. IDEF Modeling
- B. Logical Analysis via Knowledge Evaluation (LAKE)
- C. Time-Line Stress Analysis (TLSA)
- D. Behavioral Analysis

Submit

Derived Technical Requirements

Activity: Allocate Technical Requirements to Logical Analysis Models to Form Derived Technical Requirements

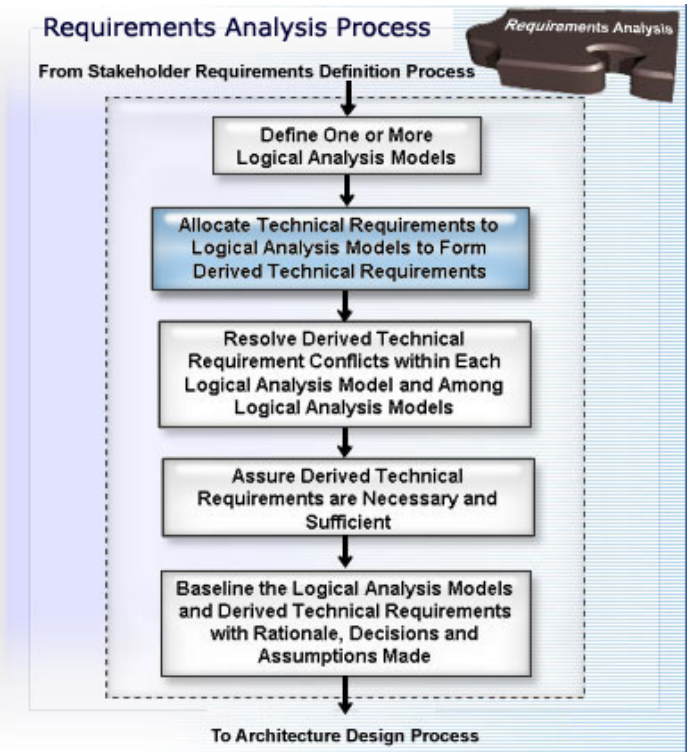
Tasks:

1. **Allocate**: Allocate performance requirements and constraints from baselined Technical Requirements to Logical Analysis Models (e.g., sub-functions, timelines, objects, data structures, IDEF diagrams).

2. **ID & Define**: Identify Derived Technical Requirements for describing functional and performance requirements, time requirements, service and attribute requirements, data flow requirements, etc., as appropriate.

3. **Acceptable?** Check that the Derived Technical Requirements are stated in an acceptable manner.

Select NEXT to continue.



D

D-Link Text:

Long Description:

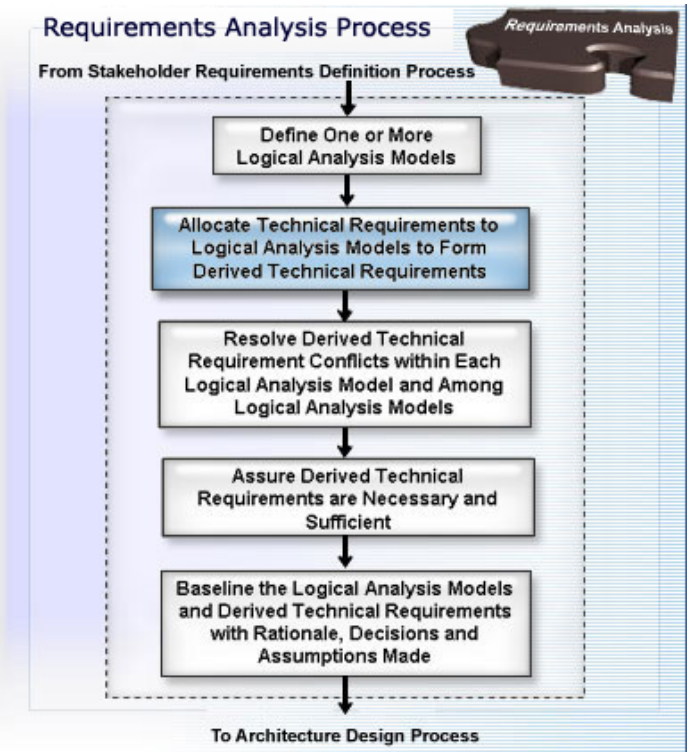
Flow begins with the Stakeholder Requirements Definition Process. The first step is to define one or more logical analysis models, then allocate technical requirements to logical analysis models to form derived technical requirements (highlighted), then resolve derived technical requirement conflicts within each logical analysis model and among logical analysis models. Then assure derived technical requirements are necessary and sufficient, then baseline the logical analysis models and derived technical requirements with rationale, decisions and assumptions made. The flow then goes to Architecture Design process.

Close window to continue.

Derived Technical Requirements, Cont.

Expected Outcomes:

- Technical Requirements (including constraints and performance requirements related to a functional requirement) that are assigned to appropriate sub-functions, timelines, objects, data structures, etc.
- Derived Technical Requirements that:
  - Reflect requirements associated with defined Logical Analysis Models
  - Represent Technical Requirements statements (e.g., range) which are not appropriate for Logical Analysis Models but through analysis can be made more specific (e.g., fuel capacity, engine efficiency and vehicle resistance)
  - Are stated in an 'acceptable' or quality manner



Select NEXT to continue.

D

D-Link Text:

Long Description:

Flow begins with the Stakeholder Requirements Definition Process. The first step is to define one or more logical analysis models, then allocate technical requirements to logical analysis models to form derived technical requirements (highlighted), then resolve derived technical requirement conflicts within each logical analysis model and among logical analysis models. Then assure derived technical requirements are necessary and sufficient, then baseline the logical analysis models and derived technical requirements with rationale, decisions and assumptions made. The flow then goes to Architecture Design process.

Close window to continue.

Acceptable Derived Technical Requirements?

A critical expected outcome of the 'Derived Technical Requirements' activity is that the derived technical requirements be stated in an acceptable manner. Obviously, this is because bad requirements generate bad designs, which ultimately result in bad systems! Over time, a variety of quality attributes or '-ilities' for a requirement have been developed via many painful lessons learned. A compilation of them includes:

- **Clarity:** Readily understandable without detailed analysis
- **Correctness**: Does not contain an error of fact
- **Feasibility**: Can be satisfied within (1) natural physical constraints, (2) state-of-the-art applicability to the project, and (3) all other project constraints
- **Focus**: Expressed in terms of 'what' and 'why,' or form, fit and function, not in detailed terms of how to develop the products or the materials to be used
- **Implementability**: Contains information necessary to enable the requirement to be implemented, but is not prescriptive in how to implement the requirement
- **Modifiability**: If needed, changes can be made completely and consistently so that upward and downward traceability can be assured.
- **Singularity**: Cannot be sensibly expressed as two or more requirements having different agents, actions, objects or instruments
- **Testability**: Existence of finite and objective procedures by which to verify that the requirement has been satisfied
- **Unambiguous**: Allows only one interpretation for meaning (e.g., not defined by terms like 'excessive,' 'sufficient' or 'resistant' that cannot be measured)
- **Verifiability**: Can be verified at the location of the system structure for which it is stated

Select NEXT to continue.

Acceptablility Among Requirements

Acceptability criteria in terms of Derived Technical Requirements also encompass how they all relate to each other as a unified set.

These types of acceptability criteria include:

- **No Redundancy**: Each requirement should be specified only once.
- **Connectivity**: All terms within a requirement are adequately linked to other requirements and to any needed word and term definitions, so those individual requirements relate properly and consistently to other requirements as a set.
- **Non-Conflicting**: A requirement is not in conflict with other ones or with itself.

Select NEXT to continue.

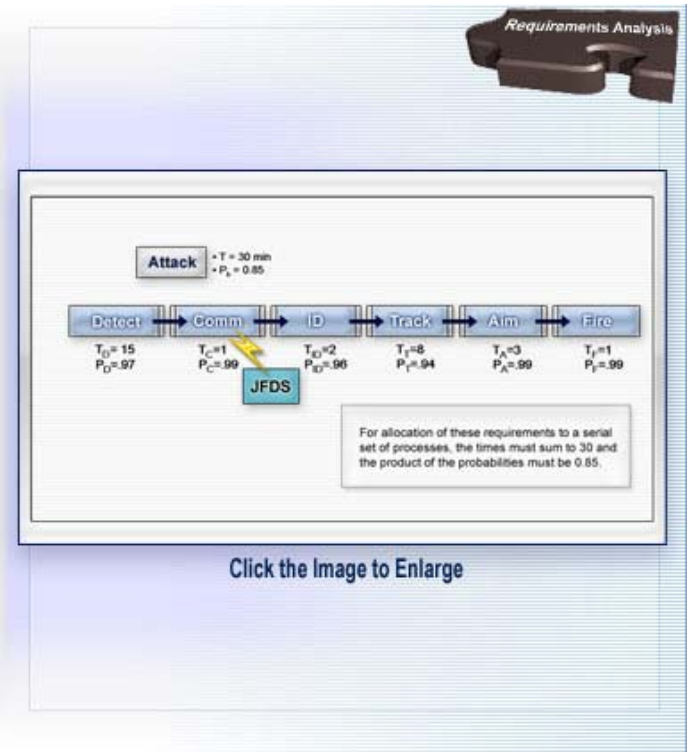
Derived Technical Requirements

To generate Derived Technical Requirements, system-level performance requirements are allocated to each sub-function. Some of the system requirements allocated may be Derived Requirements in the sense that they lack a parent requirement, but the need for their performance is ultimately derivable from some parent requirement.

The graphic elaborates the 'Attack' functional decomposition discussed earlier, but with derived performance requirements assigned.

It illustrates the allocation of time as an additive attribute to each of the sub-functions. The graphic also shows the required Probability of Kill of the system-level function and the allocated contribution by each of the sub-functions.

Select NEXT to continue.



D

D-Link Text:

Long Description:

Flowchart boxes beginning with Attack, and then which flow from Detect, Comm, ID, Track, Aim, Fire. At the bottom right is a text box which reads: For allocation of these requirements to a serial set of processes, the times must sum to 30 and the product of the probabilities must be 0.85.

Close window to continue.

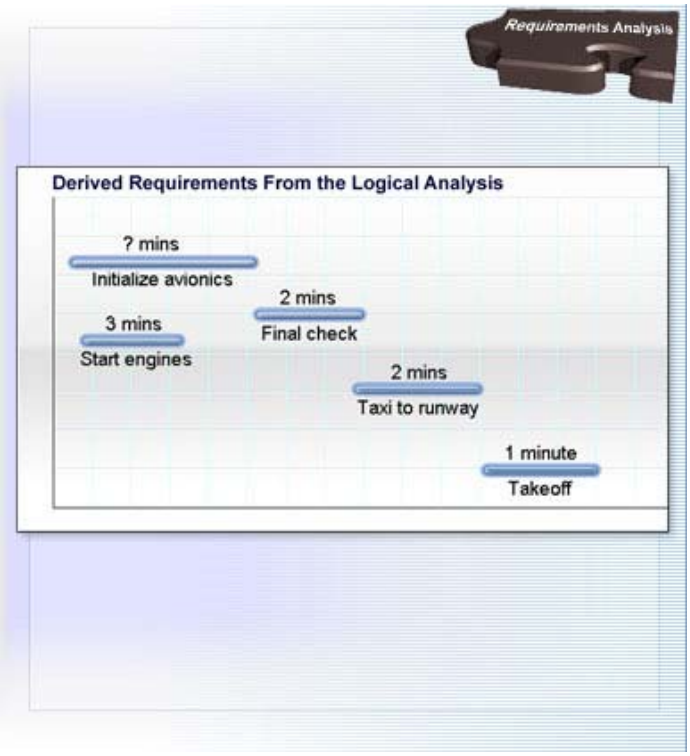
Derived Technical Requirements Illustrated

Given this mission timeline and a Technical Requirement, derived from a Key Performance Parameter (KPP), which states: "The aircraft shall be airborne within 10 minutes of an alert launch order", what is the maximum time allowable for initializing the avionics?

This initialization function is part of the system mission critical path (i.e., the longest distance from start to finish through the mission functions). The other functions take 2 + 2 + 1 = 5 minutes to complete.

Therefore, the maximum time that the avionics systems can take to initialize without impacting the total allocated mission completion time of 10 minutes is 10 - 5 = 5 minutes.

Select NEXT to continue.



D

D-Link Text:

Long Description:

A graphic titled 'Derived Requirements from the Logical Analysis'. Initialize avionics, ? minutes. Start engines, 3 minutes. Final check, 2 minutes. Taxi to runway, 2 minutes. Takeoff, 1 minute.

Close window to continue.

Popup Text:

Key Performance Parameter (KPP)
A Key Performance Parameter (KPP) identifies those key requirements that the system, End Product and Enabling Product elements must meet to be acceptable to the end user.

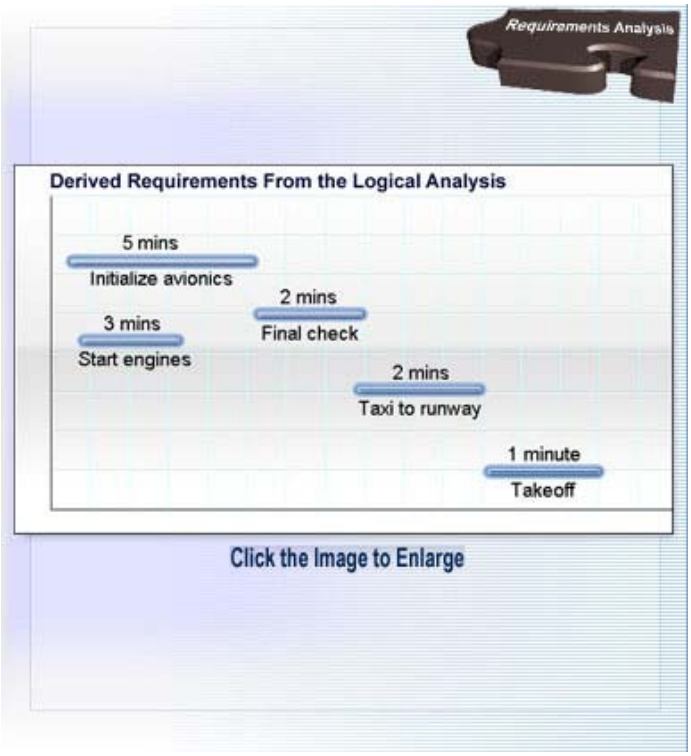
Derived Technical Requirements Illustrated, Cont.

This is a simple example of a Derived Technical Requirement identified as a result of Requirements Analysis.

Nowhere has the user explicitly stated that the avionics must be initialized in 5 minutes, but this 'derived' technical requirement nevertheless must be met in order to meet the higher-level system requirement. Failing that, the mission-level requirement must be relaxed or completion times of other functions reduced.

This Derived Technical Requirement has design solution implications. It constrains the technical design of the avionics system. It may also have impacts on cost (may be more expensive to develop such a system than planned) or schedule (more extensive development may be required than originally anticipated). Or it may require revisiting the Stakeholders Requirements Definition Process for resolution (i.e., process iteration).

Select NEXT to continue.



D

D-Link Text:

Long Description:

A graphic titled 'Derived Requirements from the Logical Analysis'. Initialize avionics, 5 minutes. Start engines, 3 minutes. Final check, 2 minutes. Taxi to runway, 2 minutes. Takeoff, 1 minute.

Close window to continue.

Knowledge Review

Please select a correct answer.

Why is the identification of Derived Technical Requirements so important?

- A. They can constrain the technical design.
- B. They can impact cost and schedule.
- C. They may prove infeasible and unless resolved, put development of the system in jeopardy.
- D. They must be known to create a complete design solution.
- E. All of the above.

Submit

Resolve Conflicts

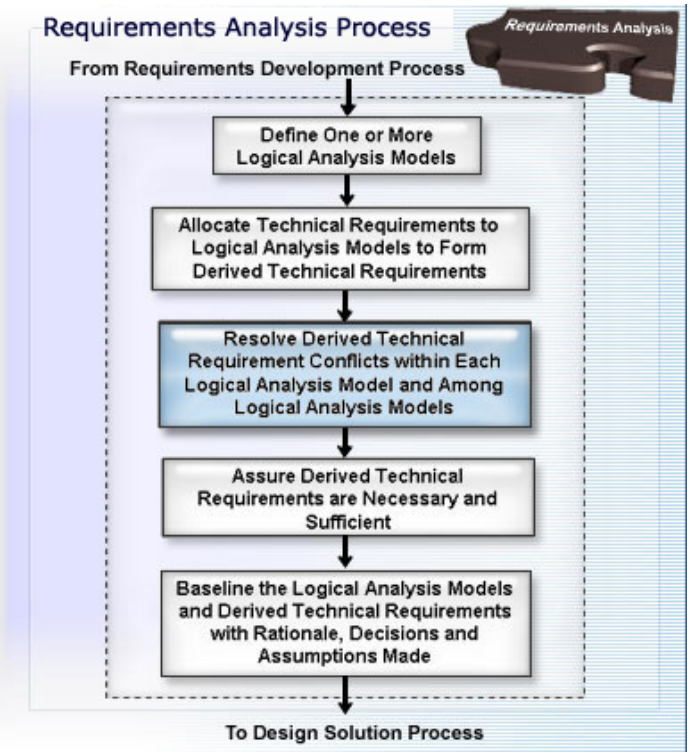
Activity: Resolve Derived Technical Requirements conflicts within each Logical Analysis Model and among Logical Analysis Models

Tasks:

1. **Model Conflicts?** Identify conflicts within each Logical Analysis Model, if any.
2. **Req't Conflicts?** Identify conflicts within Derived Technical Requirements.
3. **Trade-Off Criteria**: Use the established set of risk, cost, schedule and performance criteria in planning trade-off analyses for conflict resolution.
4. **Resolve Conflicts**: Resolve conflicts via Decision Analysis Process activities.

Expected Outcome: Conflicts are resolved.

Select NEXT to continue.



D

D-Link Text:

Long Description:

Flow begins with the Stakeholder Requirements Definition Process. The first step is to define one or more logical analysis models, then allocate technical requirements to logical analysis models to form derived technical requirements, then resolve derived technical requirement conflicts within each logical analysis model and among logical analysis models (highlighted). Then assure derived technical requirements are necessary and sufficient, then baseline the logical analysis models and derived technical requirements with rationale, decisions and assumptions made. The flow then goes to Architecture Design process.

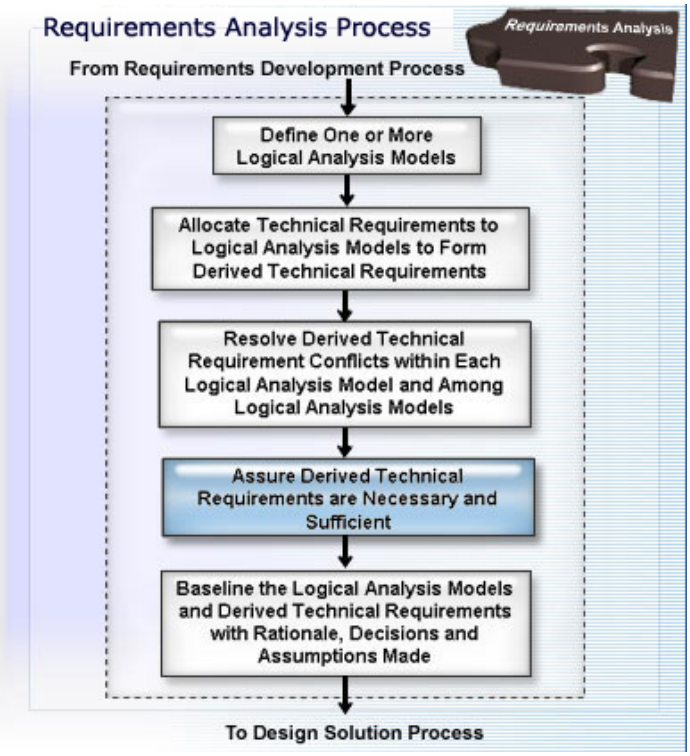
Close window to continue.

Bi-Directional Traceability

Activity: Assure Derived Technical Requirements are Necessary and Sufficient

Tasks:

1. **Trace Up?** Check upward traceability of Derived Technical Requirements from each Logical Analysis Model to its source set of Technical Requirements.
2. **Trace Down?** Check the downward traceability of Technical Requirements to their Derived Technical Requirements from each set of Logical Analysis Models.
3. **Assumptions Still OK?** Check that any assumptions and decisions made in forming Logical Analysis Models and the related set of Derived Technical Requirements are consistent with source set of Technical Requirements.
4. **Resolve Anomalies**: Resolve any anomalies identified in the above tasks.



Select NEXT to continue.

D

D-Link Text:

Long Description:

Flow begins with the Stakeholder Requirements Definition Process. The first step is to define one or more logical analysis models, then allocate technical requirements to logical analysis models to form derived technical requirements, then resolve derived technical requirement conflicts within each logical analysis model and among logical analysis models. Then assure derived technical requirements are necessary and sufficient (highlighted), then baseline the logical analysis models and derived technical requirements with rationale, decisions and assumptions made. The flow then goes to Architecture Design process.

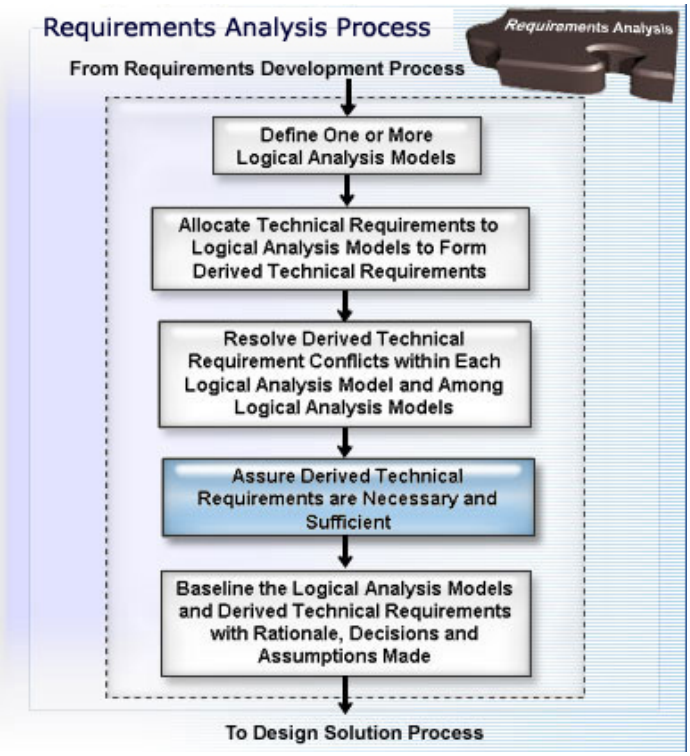
Close window to continue.

Bi-Directional Traceability, Cont.

Expected Outcomes:

- Validated sets of Derived Technical Requirements established
- Validated sets of logical models established
- Confirmation that assumptions and decisions are valid
- Resolution of identified voids, variances and conflicts

Select NEXT to continue.



D

D-Link Text:

Long Description:

Flow begins with the Stakeholder Requirements Definition Process. The first step is to define one or more logical analysis models, then allocate technical requirements to logical analysis models to form derived technical requirements, then resolve derived technical requirement conflicts within each logical analysis model and among logical analysis models. Then assure derived technical requirements are necessary and sufficient (highlighted), then baseline the logical analysis models and derived technical requirements with rationale, decisions and assumptions made. The flow then goes to Architecture Design process.

Close window to continue.

Requirements Traceability Matrix

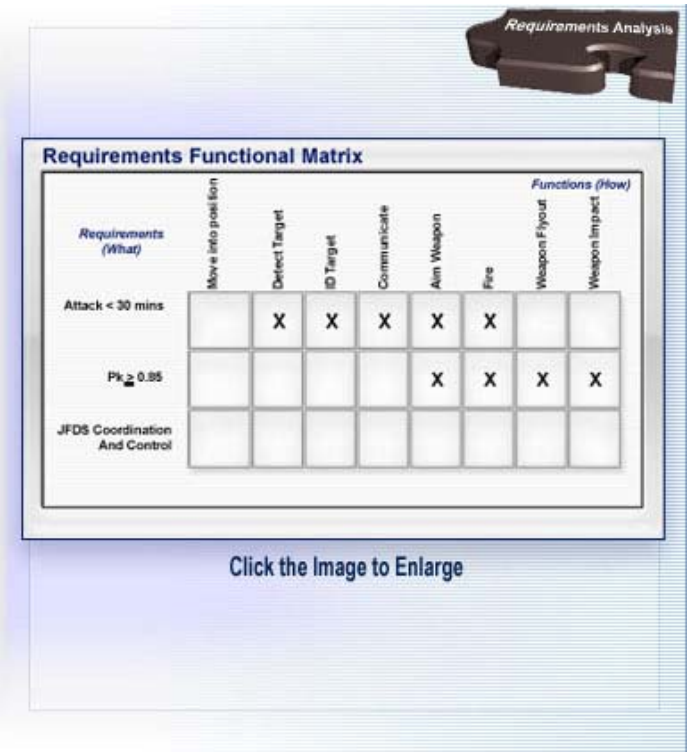
As the functional architecture is developed, a traceability matrix should be used. This matrix relates the sub-functions that resulted from decomposing system-level functions to their parent Technical Requirements (function and performance). Such a technique helps ensure traceability throughout the Requirements Analysis Process.

Each sub-function (and any resulting Derived Technical Requirements) should be checked to ensure: (1) traceability back to a system Technical Requirement, and (2) each requirement is implemented through at least one function.

This bi-directional traceability ensures each requirement is 'necessary and sufficient.'

Similar types of traceability schemes are important for both the Stakeholder Requirements Definition Process and the Architecture Design Process as well.

Select NEXT to continue.



D

D-Link Text:

Long Description:

A graphic titled 'Requirements Functional Matrix'. There are three rows under Requirements: Attack in less than 30 minutes; Pk greater than or equal to 0.85; JFDS Coordination and Control. Horizontally, there are eight functions: Move into position, Detect Target, ID Target, Communicate, Aim Weapon, Fire, Weapon Flyout and Weapon Impact. For the requirement to attack in less than 30 minutes, the functions are detect target, ID target, communicate, aim weapon and fire. For Pk greater than or equal to 0.85, the functions are aim weapon, fire, weapon flyout and weapon impact.

Close window to continue.

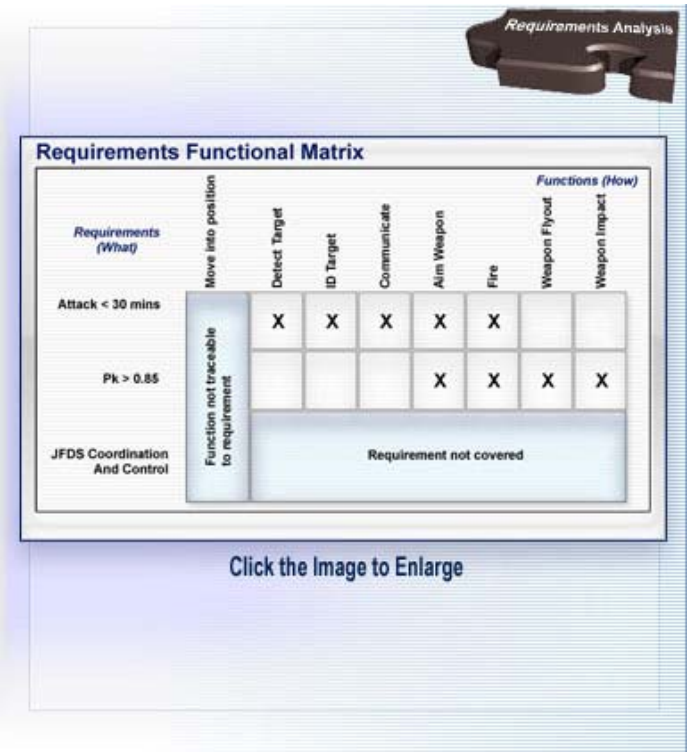
Requirements Traceability Matrix, Cont.

If there is no linkage to a requirement, then the designer has added functions and/or requirements that the user has not requested (gold-plating).

Each system Technical Requirement should trace down to at least one function.

If there are requirements with no functions, then the designers have not implemented all the needed Technical Requirements, and the system will not satisfy them. Because of that, it will ultimately fail to meet Stakeholder Requirements.

Select NEXT to continue.



D

D-Link Text:

Long Description:

A graphic titled 'Requirements Functional Matrix'. There are three rows under Requirements: Attack in less than 30 minutes; Pk greater than or equal to 0.85; JFDS Coordination and Control. Horizontally there are eight functions: Move into position, Detect Target, ID Target, Communicate, Aim Weapon, Fire, Weapon Flyout and Weapon Impact. Regarding all three requirements, moving into position function is not traceable to requirement. For the requirement to attack in less than 30 minutes, the functions are detect target, ID target, communicate, aim weapon and fire. For Pk greater than or equal to 0.85, the functions are aim weapon, fire, weapon flyout and weapon impact. For JFDS coordination and control, the requirement is not covered.

Close window to continue.

Knowledge Review

Given the requirements traceability matrix, select the correct answer.

- A. The matrix shows that all requirements are covered by at least one function.
- B. The matrix shows that 'ID Target' is not traceable to a requirement.
- C. The matrix shows that the 'JFDS Coordination and Control' is not covered by a function.
- D. Both A and B are true.

Submit

D

D-Link Text:

Long Description:

A graphic titled 'Review'. There are three rows under Requirements: Attack in less than 30 minutes; Pk greater than or equal to 0.85; JFDS Coordination and Control. Horizontally there are eight functions: Move into position, Detect Target, ID Target, Communicate, Aim Weapon, Fire, Weapon Flyout and Weapon Impact. For the requirement to attack in less than 30 minutes, the functions are Move into position, Detect target, Communicate, Aim weapon and Fire. For Pk greater than or equal to 0.85, the functions are aim weapon, fire, weapon flyout and weapon impact. For JFDS coordination and control, the function is communicate.

Close window to continue.

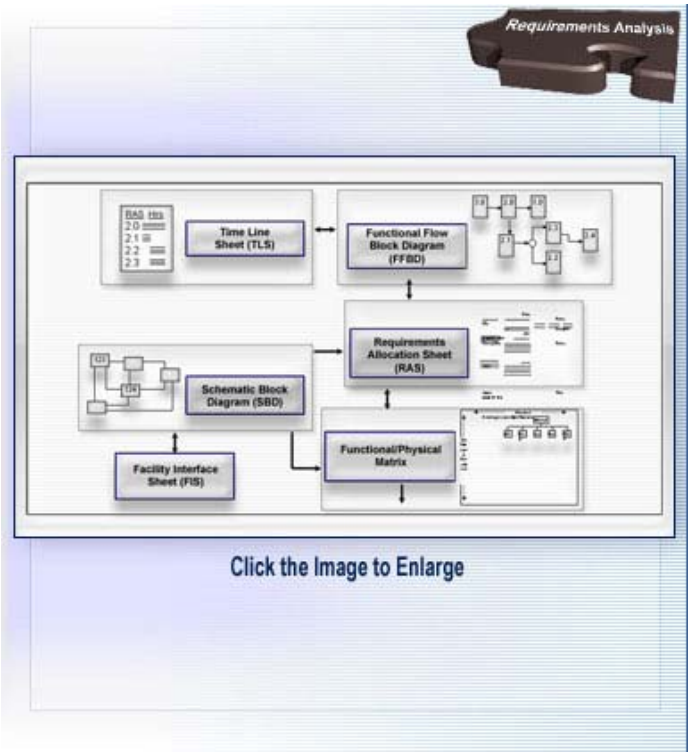
Baselined Outputs

In addition to baselining Stakeholder Requirements and Technical Requirements, the outputs of the Requirements Analysis Process are also baselined.

This baselining is important since these outputs are used to develop many supporting and enabling products for using the system.

They can also be used to address future threats in order to determine if changes in the way the system is functionally employed can be used to counter a new threat or emerging technology.

Select NEXT to continue.



D

D-Link Text:

Long Description:

A flowchart beginning with Time Line Sheet, then Functional Flow Block Diagram, Requirements Allocation Sheet, Functional/Physical Matrix. Another flow begins with Schematic Block Diagram and can flow to one of three areas: Requirements Allocation Sheet, Facility Interface Sheet and Functional/Physical Matrix.

Close window to continue.

Capture Logical Analysis Work Products

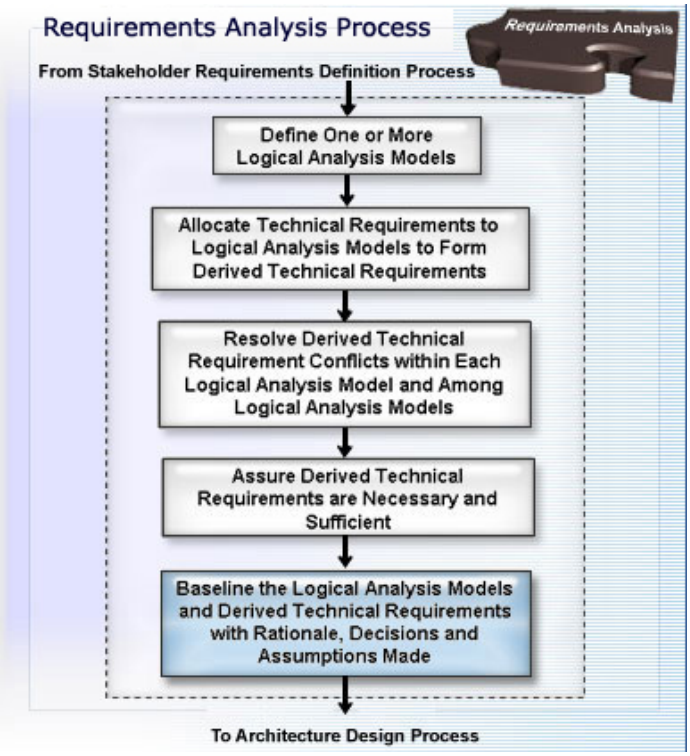
Activity: Baseline the Logical Analysis Models and Derived Technical Requirements with decisions, decision rationale and assumptions made

Tasks:

1. **Capture Key Info**: Capture Logical Analysis Models and Derived Technical Requirements.
2. **Record It**: Record logical analysis work products and Derived Technical Requirements in the established technical data management database so as to permit requisite traceability.
3. **Get Lessons Learned**: Capture and record lessons learned from applying the Requirements Analysis Process in the established technical data management information database.

Expected Outcome: Requirements Analysis work products are captured and baselined.

Select NEXT to continue.



D

D-Link Text:

Long Description:

Flow begins with the Stakeholder Requirements Definition Process. The first step is to define one or more logical analysis models, then allocate technical requirements to logical analysis models to form derived technical requirements, then resolve derived technical requirement conflicts within each logical analysis model and among logical analysis models. Then assure derived technical requirements are necessary and sufficient, then baseline the logical analysis models and derived technical requirements with rationale, decisions and assumptions made (highlighted). The flow then goes to Architecture Design process.

Close window to continue.

Role of QFD

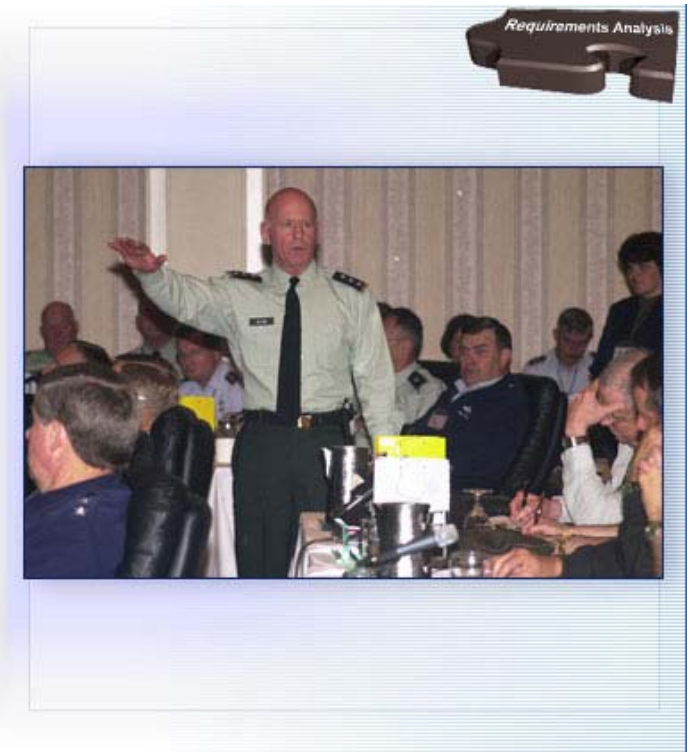
Within the DoD acquisition system, a number of specially-designated working-level IPTs (WIPTs) are used to provide feedback and guidance to programs.

Additionally, as part of the Technical Processes involved in top-down design, the need for close coordination with the end user and system stakeholders is essential as various trade-offs are made during the Requirements Analysis Process.

There are many techniques to ensure that such feedback and guidance are relevant, effective and involve all key players. Otherwise, much of the work involved in Requirements Analysis, which directly influences the design solution, may be wasted.

One approach to assist in this effort is called Quality Function Deployment (QFD). Developed in Japan in the 1960s, QFD began being used in US industry in the late 1980's.

Select NEXT to continue.



What is QFD?

QFD is a disciplined way of translating customer needs into detailed product requirements which can be deployed to influence system design and other product attributes.

QFD covers parts of Stakeholder Requirements Definition, Requirements Analysis and Architecture Design activities. QFD is an enabler for developing Stakeholder Requirements (via the "voice of the customer"), translating requirements into functions and those functions into acceptable solutions.

While not a substitute for applying the full set of system design processes, QFD is a useful approach that can be used to supplement.

From that perspective, QFD's use is illustrated on the following screens.

Select NEXT to continue.

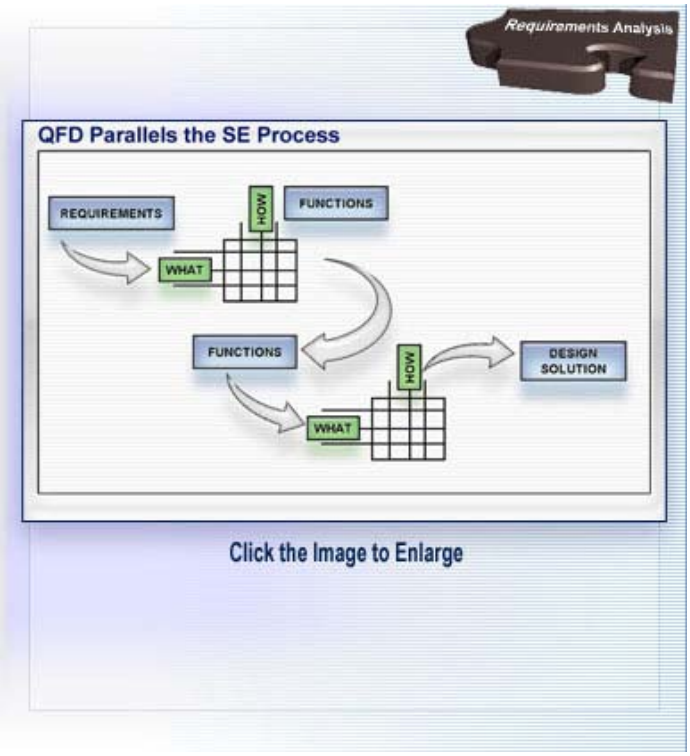
QFD: 'Voice of the Customer'

QFD starts with 'the voice of the customer'. It uses a series of matrices to 'flow down' requirements to various levels. An example is illustrated here.

In this example, the 'what' is shown in the left side of the matrix. Using various QFD processes, this ultimately has led to development of the 'how' or 'what' functions might be accomplished.

The 'how's' of this matrix then become the what's of the next lower matrix.

Select NEXT to continue.



D

D-Link Text:

Long Description:

The chart is titled “QFD Parallels the SE Process”. It shows Requirements feeding into “What”. Then Functions feeding into “How”. These Functions then feed into “What” and “How” to produce a Design Solution.

Close window to continue.

QFD Scoring

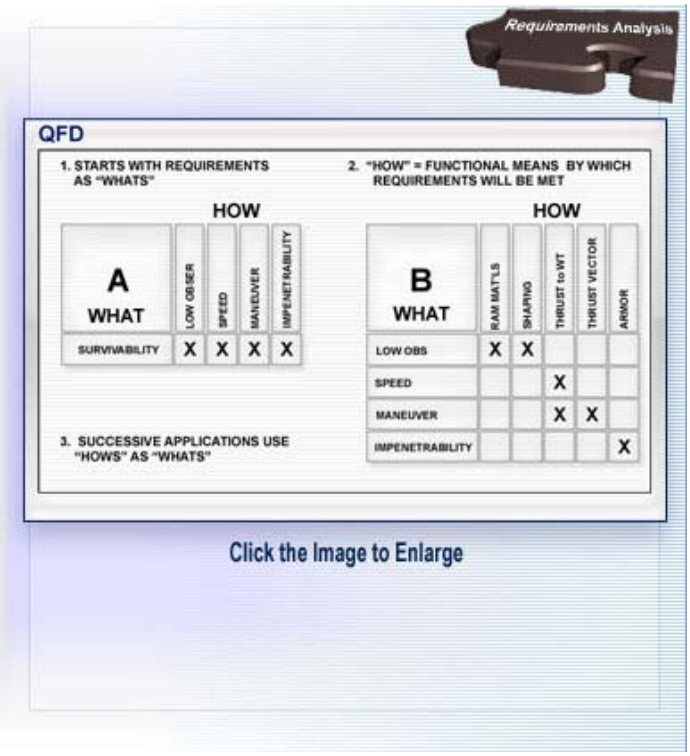
QFD allows experts from various fields to score how well the 'how's' either support or detract from the 'what's'.

In the example to the right, design experts have identified the impact of each of the design options to support achievement of different functions. The functions were derived from the stakeholder's requirement for survivability.

In this example, performance aspects of various design options are being evaluated.

The cost implications, reliability and maintainability, producibility and any other important program aspects can also be evaluated in separate matrices.

Select NEXT to continue.



D

D-Link Text:

Long Description:

Chart titled 'QFD'. 1. Starts with requirements as 'whats.' In chart A, 'what' is survivability. 'How' is low observation, speed, maneuverability and impenetrability. 2. 'How' equals functional means by which requirements will be met. In B, there are several 'whats.' Low observation by RAM materials and shaping. Speed by thrust to WT. Maneuver by thrust to WT and thrust vector. Impenetrability by armor. 3. Successive applications use 'hows' and 'whats.'

Close window to continue.

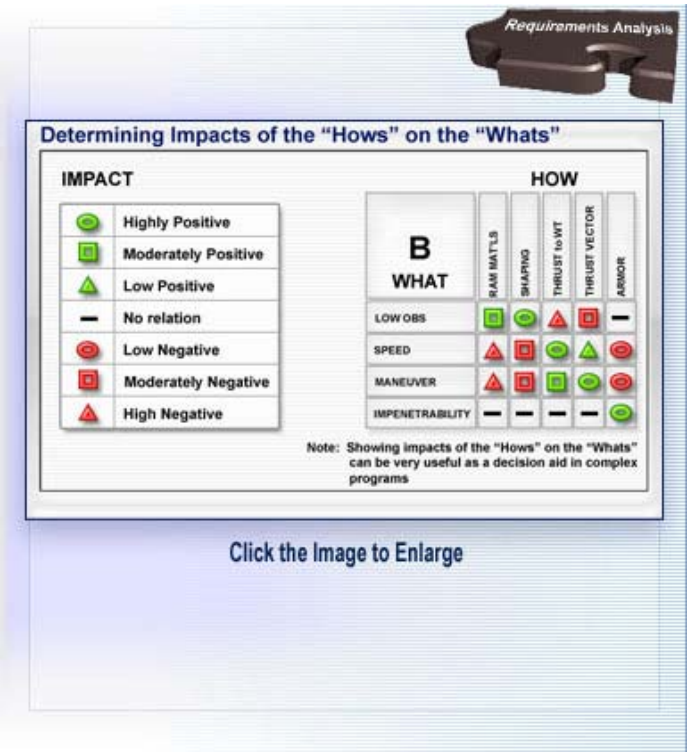
Determining Impacts

QFD provides a method to prioritize different options.

In this example, design experts were asked to rate how each of the design options ('how') either enhanced or detracted from achievement of the functions ('what').

Their responses are shown here.

Select NEXT to continue.



D

D-Link Text:

Long Description:

Graphic titled 'Determining Impacts of the 'Hows' on the 'Whats'. There is a guide at the left representing Impact. A green circle is highly positive. A green square is moderately positive. A green triangle is low positive. A dash means no relation. A red circle is low negative. A red square is moderately negative. A red triangle is high negative. There is a chart to the right--the 'whats' are low observation, speed, maneuver and impenetrability. The 'hows' are RAM materials, shaping, thrust to WT, thrust vector and armor. For low observation, the RAM materials are moderately positive. The shaping is highly positive. The thrust to WT is highly negative. The thrust vector is moderately negative, and the armor is no relation. Regarding speed, the RAM materials are highly negative. The shaping is moderately negative. The thrust to WT is highly positive. The thrust vector is low positive, and the armor is low negative. Regarding maneuver, the RAM materials are highly negative. The shaping is moderately negative. The thrust to WT is moderately positive. The thrust vector is highly positive, and the armor is low negative. For impenetrability, all the 'hows' are no relation, except armor, which is highly positive.

Close window to continue.

Quantifying User Needs

The users were then asked to rate how well each of the functions contributed to the achievement of survivability.

The results were normalized so that all of the 'what' contributions sum to one.

To quantify the design options, each user priority is multiplied by the level of impact score assigned by the design experts.

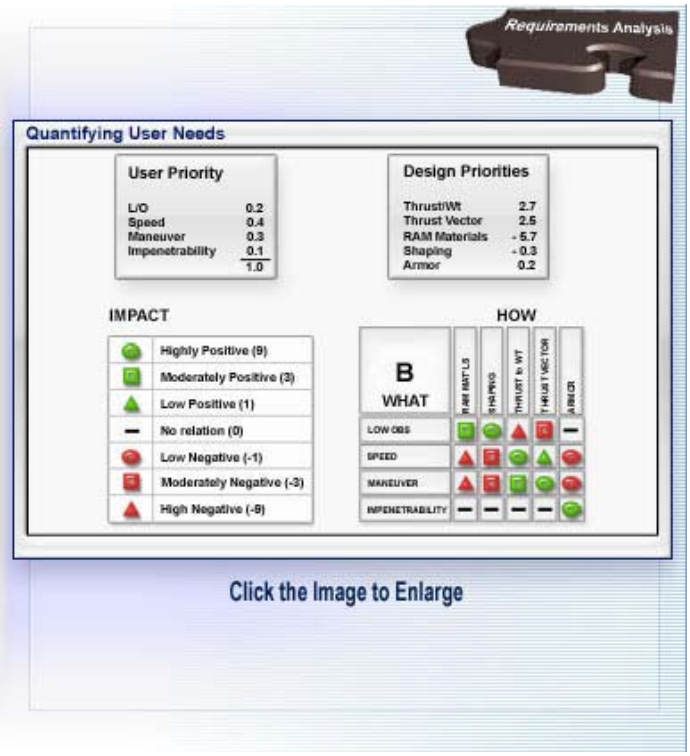
The scores of each column are then summed to determine the quantitative impact of each design option for meeting the requirement of survivability.

For example, the Design Priority for 'RAM Materials' is computed as:

RAM SCORE = (3 x .2) + (-9 x .4) + (-9 x .3)

RAM SCORE = -5.7

Select NEXT to continue.



D

D-Link Text:

Long Description:

Graphic titled 'Quantifying User Needs'. There are two text boxes in the upper part of the graphic. The upper left box is titled 'User Priority.' L/O. is 0.2. Speed is 0.4. Maneuver is 0.3. Impenetrability is 0.1. The sum is 1.0. The box on the upper right is titled 'Design Priorities.' Thrust/Wt is 2.7. Thrust Vector is 2.5. RAM Materials is -5.7. Shaping is -0.3. Armor is 0.2. At the lower left is a chart titled 'Impact.' A green circle is highly positive (9). A green square is moderately positive (3). A green triangle is low positive (1). A dash means no relation (0). A red circle is low negative (-1). A red square is moderately negative (-3). A red triangle is high negative (-9). There is a chart to the right--the 'whats' are low observation, speed, maneuver and impenetrability. The 'hows' are RAM materials, shaping, thrust to WT, thrust vector and armor. For low observation, the RAM materials are moderately positive. The shaping is highly positive. The thrust to WT is highly negative. The thrust vector is moderately negative, and the armor is no relation. Regarding speed, the RAM materials are highly negative. The shaping is moderately negative. The thrust to WT is highly positive. The thrust vector is low positive, and the armor is low negative. Regarding maneuver, the RAM materials are highly negative. The shaping is moderately negative. The thrust to WT is moderately positive. The thrust vector is highly positive, and the armor is low negative. For impenetrability, all the 'hows' are no relation, except armor, which is highly positive.

Close window to continue.

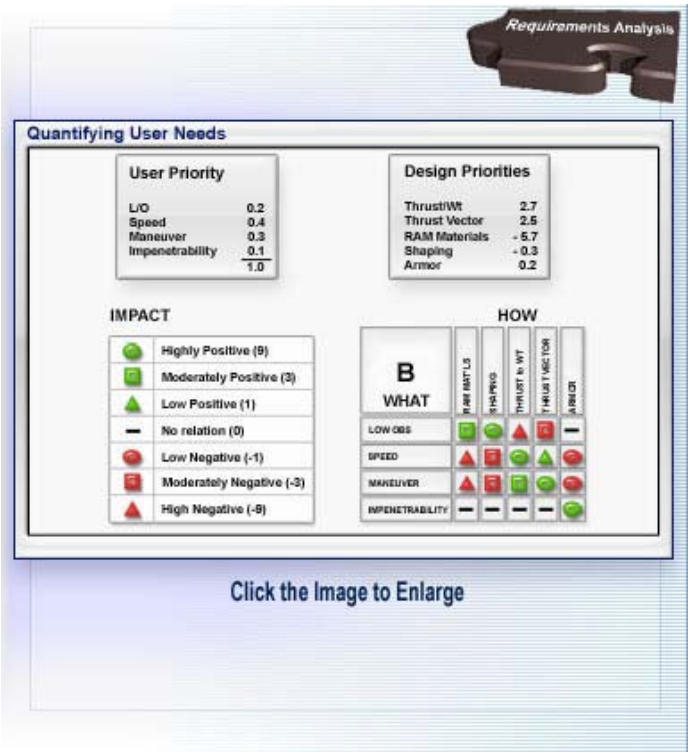
Design Evolution

The QFD process can be used through each level as the design becomes increasingly more detailed. Different QFD matrices (e.g., performance, supportability, cost, etc.) can also be combined to create a composite program score.

There are commercial software applications which automate the calculations of the QFD process.

The result is a design evolution with a well documented design process that maintains traceability back to the original customer inputs.

Select NEXT to continue.



D

D-Link Text:

Long Description:

Graphic titled 'Quantifying User Needs'. There are two text boxes in the upper part of the graphic. The upper left box is titled 'User Priority.' L/O. is 0.2. Speed is 0.4. Maneuver is 0.3. Impenetrability is 0.1. The sum is 1.0. The box on the upper right is titled 'Design Priorities.' Thrust/Wt is 2.7. Thrust Vector is 2.5. RAM Materials is -5.7. Shaping is -0.3. Armor is 0.2. At the lower left is a chart titled 'Impact.' A green circle is highly positive (9). A green square is moderately positive (3). A green triangle is low positive (1). A dash means no relation (0). A red circle is low negative (-1). A red square is moderately negative (-3). A red triangle is high negative (-9). There is a chart to the right--the 'whats' are low observation, speed, maneuver and impenetrability. The 'hows' are RAM materials, shaping, thrust to WT, thrust vector and armor. For low observation, the RAM materials are moderately positive. The shaping is highly positive. The thrust to WT is highly negative. The thrust vector is moderately negative, and the armor is no relation. Regarding speed, the RAM materials are highly negative. The shaping is moderately negative. The thrust to WT is highly positive. The thrust vector is low positive, and the armor is low negative. Regarding maneuver, the RAM materials are highly negative. The shaping is moderately negative. The thrust to WT is moderately positive. The thrust vector is highly positive, and the armor is low negative. For impenetrability, all the 'hows' are no relation, except armor, which is highly positive.

Close window to continue.

Knowledge Review

Please select a correct answer.

All of the following statements about QFD are true **except** _____.

- A. QFD starts with the 'voice of the customer'.
- B. QFD is an analytical method to prioritize options.
- C. QFD is a substitute for Systems Engineering Technical Processes.
- D. QFD is a way to gain consensus among diverse groups.

Submit

Review of Objectives

Requirements Analysis: Yields sets of Logical Analysis Models (functional architectures) and related Derived Technical Requirements that improve understanding of the Technical Requirements and the scope of the problem to be solved by the Architecture Design Process

- **Key Inputs**: The Technical Requirements from the Stakeholder Requirements Definition Process
- **Key Outputs**: The baselined Logical Analysis Models making up the functional architecture and Derived Technical Requirements. These major outputs of Requirements Analysis become inputs to the Architecture Design Process

Requirements Analysis Process Activities: Define Logical Analysis Models; Allocate Technical Requirements to Logical Analysis Models to form Derived Technical Requirements; Resolve Derived Technical Requirements conflicts with each Logical Analysis Model and among Logical Analysis Models; Assure that Derived Technical Requirements are necessary and sufficient (exhibit bi-directional traceability with the Technical Requirements); Baseline the Logical Analysis Models and Derived Technical Requirements with decisions made, decision rationale and assumptions

Process Relationships: Requirements Analysis complements the Technical Processes of Stakeholder Requirements Definition and Architecture Design. This complementary relationship is important since some requirements will become fully defined only through system decomposition at later stages of development. These three processes are repeated on each system model from the top-down of the system structure during each phase of the life cycle.

Select NEXT to continue.

Review of Objectives, Cont.

Types of Requirements

- **Derived Requirements**: A requirement that is not explicitly stated in the set of Stakeholder Requirements yet is required to satisfy one or more specific Stakeholder Requirements. Many of these are addressed as part of the Stakeholder Requirements Definition process.

- **Derived Technical Requirements**: A requirement that results from a Requirements Analysis decision or physical Architecture Design alternative analysis decision. These types of derived technical requirements are addressed as a part of Requirements Analysis and Design activities.

Functional Architecture: Is the collection of logical models produced during Requirements Analysis. The benefits of a functional architecture include being able to better measure fulfillment of Stakeholder Requirements; assessment of system's ability to operate within resource constraints; an understanding of life cycle costs; and providing a basis for trade-offs.

Quality Function Deployment (QFD): Not a substitute for Systems Engineering processes, QFD is a disciplined, multifunctional team-based approach. QFD is a disciplined way of translating customer needs into detailed product requirements, which can be deployed to influence system design and other product attributes.

Select NEXT to continue.

Review of Objectives, Cont.

Tools and Techniques: A wide variety of tools and techniques can be used in Requirements Analysis. These include:

- **Functional Analysis**: Each system function is decomposed to its needed set of sub-functions (also described by subject-verb-object statements) to carry out the system function.
- **Functional Flow Block Diagram**: A logical arrangement of the functions and sub-functions. By analyzing those that can be done in serial sequence or in parallel, interfaces and interactions of functions and sub-functions can be better understood.
- **N2 Chart**: Can be used to help define system interfaces. The N2 Chart complements the Functional Flow Block Diagram and highlights the data flows as inputs and outputs of system functions.
- **Timeline Analysis**: This analysis defines the duration and sequence of various time-critical functions.
- **Time Line Sheet**: Provides a form for showing the relationship of functions based on time. It shows concurrency, overlapping and sequential relationships of functions.
- **Structured Analysis**: This tool is used by software projects. It models the data flows and functional flows and provides three views of the problem: an Activity View, Data View and Rule View.

Select NEXT to continue.

Review of Objectives, Cont.

Tools and Techniques (cont):

- **Object-Oriented Analysis**: Is primarily used by software developers. This approach uses three views of an object from which the services and attributes of the object can be derived. This approach is particularly useful for software reuse opportunities.
- **IDEF Diagrams**: Are used for information and business systems that need to model data flows. The diagrams can be used to graphically depict structured representations of activities and functions as well as data flows.
- **UML and SysML**: UML is a general-purpose graphical modeling language used to create abstract models of a system. The Systems Modeling Language (SysML) is an extension of a subset of UML. SysML supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems.
- **Behavioral Analysis**: Involves the simulation/stimulation of functional architectures utilizing operational scenarios to expose the model to stressful situations that reflect anticipated usage and environments. Such analysis helps to provide understanding of how a product can be expected to act to various stimuli.
- **Requirements Allocation Sheets**: Enables various aspects of a set of requirements to be captured and analyzed.
- **Requirements Traceability Matrix**: Relates sub-functions to their parent Technical Requirements.

You have reached the end of this topic. To proceed, select the exam for this topic in the Table of Contents.