The last subsection in section 3 specifies Precedence and criticality of requirements listed in the previous subsections. The order of precedence, criticality, or assigned weights indicating the relative importance of the requirements in this specification are clearly documented. Examples include identifying those requirements deemed critical to safety, to security, or to privacy for purposes of singling them out for special treatment. If all requirements have equal weight, this paragraph shall so state.

Section 4 entitled Qualification provisions defines a set of qualification methods and shall specify for each requirement in Section 3 the method(s) to be used to ensure that the requirement has been met. A table may be used to present this information, or each requirement in Section 3 may be annotated with the method(s) to be used. Qualification methods may include:

- Demonstration: The operation of the CSCI, or a part of the CSCI, that relies on observable functional operation not requiring the use of instrumentation, special test equipment, or subsequent analysis.
- Test: The operation of the CSCI, or a part of the CSCI, using instrumentation or other special test equipment to collect data for later analysis.
- Analysis: The processing of accumulated data obtained from other qualification methods. Examples are reduction, interpretation, or extrapolation of test results.
- Inspection: The visual examination of CSCI code, documentation, etc.
- Special qualification methods: Any special qualification methods for the CSCI, such as special tools, techniques, procedures, facilities, and acceptance limits.

Finally section 5 documents the Requirements traceability information as follows:

- Traceability from each CSCI requirement in this specification to the system (or subsystem, if applicable) requirements it addresses. (Alternatively, this traceability may be provided by annotating each requirement in Section 3.)

Note: Each level of system refinement may result in requirements not directly traceable to higher-level requirements. For example, a system architectural design that creates multiple CSCIs may result in requirements about how the CSCIs will interface, even though these interfaces are not covered in system requirements. Such requirements may be traced to a general requirement such as “system implementation” or to the system design decisions that resulted in their generation.

- Traceability from each system (or subsystem, if applicable) requirement allocated to this CSCI to the CSCI requirements that address it. All system (subsystem) requirements allocated to this CSCI shall be accounted for. Those that trace to CSCI requirements contained in IRSs shall reference those IRSs.

Traceability analysis is a key activity in the verification and validation process described briefly in the previous Chapter and will be described in more detail in Chapter 6 of this book.

memory, input/output devices, auxiliary storage, communications/network equipment, and other required equipment. Examples of software resources requirements include operating systems, database management systems, communications/ network software, utility software, input and equipment simulators, test software, and manufacturing software. Examples of requirements for communication resources include geographic locations to be linked; configuration and network topology; transmission techniques; data transfer rates; gateways; required system use times; type and volume of data to be transmitted/received; time boundaries for transmission/ reception/response; peak volumes of data; and diagnostic features.

Section 3.11 entitled Software quality factors specifies the CSCI requirements, if any, concerned with software quality factors identified in the contract or derived from a higher level specification. Examples include quantitative requirements regarding CSCI functionality (the ability to perform all required functions), reliability (the ability to perform with correct, consistent results), maintainability (the ability to be easily corrected), availability (the ability to be accessed and operated when needed), flexibility (the ability to be easily adapted to changing requirements), portability (the ability to be easily modified for a new environment), reusability (the ability to be used in multiple applications), testability (the ability to be easily and thoroughly tested), usability (the ability to be easily learned and used), and other attributes.

Design and implementation constraints are specified in section 3.12. This section specifies the requirements, if any, that constrain the design and implementation of the CSCI. These requirements may be specified by reference to appropriate commercial or military standards and specifications. Examples include requirements concerning:

- Use of a particular CSCI architecture or requirements on the architecture, such as required databases or other software units; use of standard, military, or existing components; or use of Government/acquirer-furnished property (equipment, information, or software)
- Use of particular design or implementation standards; use of particular data standards; use of a particular programming language
- Flexibility and expendability that must be provided to support anticipated areas of growth or changes in technology, threat, or mission

Sections 3.13, 3.14, and 3.15 specify the Personnel-related requirements, Training-related requirements, and Logistics-related requirements, respectively. The first section shall specify the CSCI requirements, if any, included to accommodate the number, skill levels, duty cycles, training needs, or other information about the personnel who will use or support the CSCI. Examples include requirements for number of simultaneous users and for built-in help or training features. Also included shall be the human factors engineering requirements, if any, imposed on the CSCI.

Training-related requirements specify the CSCI requirements, if any, pertaining to training. Examples include training software to be included in the CSCI. Logistics-related requirements specify the CSCI requirements, if any, concerned with logistics considerations. These considerations may include: system maintenance, software support, system transportation modes, supply-system requirements, impact on existing facilities, and impact on existing equipment.

- Synchronization, including connection establishment, maintenance, termination
- Status, identification, and any other reporting features

g) Other required characteristics, such as physical compatibility of the interfacing entities (dimensions, tolerances, loads, plug compatibility, etc.), voltages, etc.

The above section, section 3.3, is considered as one of the most important sections in this document. This is because many of the errors found during testing are traced to inappropriately specified interfaces. The following section, section 3.4 in the document, describes the CSCI internal interface requirements. This section shall specify the requirements, if any, imposed on interfaces internal to the CSCI. If all internal interfaces are left to the design, this fact shall be so stated. If such requirements are to be specified, then the topics in section 3.3 above can be used also for internal interfaces.

The information specification is continued in the following section entitled CSCI internal data requirements. This section specifies the requirements, if any, imposed on data internal to the CSCI. Included shall be requirements, if any, on databases and data files to be included in the CSCI. If all decisions about internal data are left to the design, this fact shall be so stated. If such requirements are to be imposed, paragraphs 3.3.x.e and 3.3.x.d above provide a list of topics to be considered.

Adaptation requirements are specified in section 3.6. This section shall specify the requirements, if any, concerning installation-dependent data to be provided by the CSCI (such as site-dependent latitude and longitude or site-dependent state tax codes) and operational parameters that the CSCI is required to use that may vary according to operational needs (such as parameters indicating operation-dependent targeting constants or data recording).

Safety requirements are specified next. The CSCI requirements, if any, concerned with preventing or minimizing unintended hazards to personnel, property, and the physical environment must be thoroughly specified. Examples include safeguards the CSCI must provide to prevent inadvertent actions (such as accidentally issuing an “auto pilot off” command) and non-actions (such as failure to issue an intended “auto pilot off” command). This section shall include the CSCI requirements, if any, regarding nuclear components of the system, including, as applicable, prevention of inadvertent detonation and compliance with nuclear safety rules.

Security and privacy requirements is a separate section specifying the CSCI requirements, if any, concerned with maintaining security and privacy. These requirements shall include, as applicable, the security/privacy environment in which the CSCI must operate, the type and degree of security or privacy to be provided, the security/privacy risks the CSCI must withstand, required safeguards to reduce those risks, the security/privacy policy that must be met, the security/privacy accountability the CSCI must provide, and the criteria that must be met for security/privacy certification/accreditation.

A CSCI environment requirements section specifies the requirements, if any, regarding the environment in which the CSCI must operate. Examples include the computer hardware and operating system on which the CSCI must run. (Additional requirements concerning computer resources are given in the next paragraph.)

Hardware, software, and communications resources requirements are specified next in section 3.10. Examples of hardware resources requirements include characteristics of processors,

of the interfacing entities (such as different expectations about the size, frequency, or other characteristics of data elements):

a) Priority that the CSCI must assign the interface
b) Requirements on the type of interface (such as real-time data transfer, storage-and-retrieval of data, etc.) to be implemented
c) Required characteristics of individual data elements that the CSCI must provide, store, send, access, receive, etc., such as:
    - Data type (alphanumeric, integer, etc.)
    - Size and format (such as length and punctuation of a character string)
    - Units of measurement (such as meters, dollars, nanoseconds)
    - Range or enumeration of possible values (such as 0-99)
    - Accuracy (how correct) and precision (number of significant digits)
    - Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the data element may be updated and whether business rules apply.
    - Security and privacy constraints
    - Sources (setting/sending entities) and recipients (using/receiving entities)
d) Required characteristics of data element assemblies (records, messages, files, arrays, displays, reports, etc.) that the CSCI must provide, store, send, access, receive, etc., such as:
    - Data elements in the assembly and their structure (number, order, grouping)
    - Medium (such as disk) and structure of data elements/assemblies on the medium
    - Visual and auditory characteristics of displays and other outputs (such as colors, layouts, fonts, icons and other display elements, beeps, lights)
    - Relationships among assemblies, such as sorting/access characteristics
    - Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the assembly may be updated and whether business rules apply
    - Security and privacy constraints
    - Sources (setting/sending entities) and recipients (using/receiving entities)
e) Required characteristics of communication methods that the CSCI must use for the interface, such as:
    - Project-unique identifier(s)
    - Communication links/bands/frequencies/media and their characteristics
    - Message formatting
    - Flow control (such as sequence numbering and buffer allocation)
    - Data transfer rate, whether periodic/aperiodic, and interval between transfers
    - Routing, addressing, and naming conventions
    - Transmission services, including priority and grade
    - Safety/security/privacy considerations, such as encryption, user authentication, compartmentalization, and auditing
f) Required characteristics of protocols the CSCI must use for the interface, such as:
    - Project-unique identifier(s)
    - Priority/layer of the protocol
    - Packeting, including fragmentation and reassembly, routing, and addressing
    - Legality checks, error control, and recovery procedures

After specifying the states and the modes of the CSCI, the functional requirements for each state are specified. This comes under the heading CSCI capability requirements.

This section is divided into subsections to itemize the requirements associated with each capability of the CSCI. The term “capability” is defined as a group of related requirements. This is a generic term which may be replaced with “function”, “subject”, “object”, or other term useful for presenting the requirements. The subsection designated as 3.2.x (CSCI capability) identifies a required CSCI capability and itemizes the requirements associated with it. If the capability can be more clearly specified by dividing it into constituent capabilities. The constituent capabilities shall be specified in subsections. The requirements must specify required behavior of the CSCI and shall include applicable parameters, such as response times, throughput times, other timing constraints, sequencing, accuracy, capacities (how much/how many), priorities, continuous operation requirements, and allowable deviations based on operating conditions. The requirements shall include, as applicable, required behavior under unexpected, unallowed, or “out of bounds” conditions, requirements for error handling, and any provisions to be incorporated into the CSCI to provide continuity of operations in the event of emergencies. The section provides a list of topics to be considered when specifying requirements regarding inputs the CSCI must accept and must produce outputs.

Sections 3.1 and 3.2, described above contain the graphical analysis diagrams using either SA or OOA as introduced in the previous sections. This is needed in order to graphically represent the overall CSCI functional partitioning, behavioral specification, information flow representation and modeling. The CSCI external interface requirements is divided into subsections to specify the requirements, if any, for the CSCI's external interfaces. This section may reference one or more Interface Requirements Specifications (IRSs) document or other documents containing these requirements.

The first subsection, Interface identification and diagrams, identifies the required external interfaces of the CSCI (that is, relationships with other entities that involve sharing, providing or exchanging data). The identification of each interface shall include a project-unique identifier and shall designate the interfacing entities (systems, configuration items, users, etc.) by name, number, version, and documentation references, as applicable. The identification shall state which entities have fixed interface characteristics (and therefore impose interface requirements on interfacing entities) and which are being developed or modified (thus having interface requirements imposed on them). One or more interface diagrams must be provided to depict the interfaces. These diagrams are in the form of context diagrams as mentioned in the previous sections.

The second and subsequent subsections (beginning with 3.3.2) must identify a CSCI external interface by project-unique identifier, briefly identify the interfacing entities, and must be divided into subparagraphs as needed to state the requirements imposed on the CSCI to achieve the interface. Interface characteristics of the other entities involved in the interface shall be stated as assumptions or as “When [the entity not covered] does this, the CSCI shall...,” not as requirements on the other entities. This paragraph may reference other documents (such as data dictionaries, standards for communication protocols, and standards for user interfaces) in place of stating the information here.

The requirements shall include the following, as applicable, presented in any order suited to the requirements, and shall note any differences in these characteristics from the point of view

1. Scope.
    1.1 Identification
    1.2 System overview
    1.3 Document overview
2. Referenced documents.
3. Requirements
    3.1 Required states and modes
    3.2 CSCI capability requirements
        3.2.x (CSCI capability)
    3.3 CSCI external interface requirements
        3.3.1 Interface identification and diagrams
        3.3.x (Project-unique identifier of interface)
    3.4 CSCI internal interface requirements
    3.5 CSCI internal data requirements
    3.6 Adaptation requirements
    3.7 Safety requirements
    3.8 Security and privacy requirements
    3.9 CSCI environment requirements
    3.10 Computer resource requirements
        3.10.1 Computer hardware requirements
        3.10.2 Computer hardware resource utilization requirements
        3.10.3 Computer software requirements
        3.10.4 Computer communications requirements
    3.11 Software quality factors
    3.12 Design and implementation constraints
    3.13 Personnel-related requirements
    3.14 Training-related requirements
    3.15 Logistics-related requirements
    3.16 Precedence and criticality of requirements
4. Qualification provisions
5. Requirements traceability

Table 1: Outline of the Software Requirements Specification Documents

this section, in an appendix referenced from this section, or by annotation of the requirements in the paragraphs or sections where they appear.

The detailed discussion of OOA and examples of using TWK/OOA will be presented later in this Chapter.

3.1.3 The SWRA Documents

The Software Requirements Specification (SRS) document specifies the requirements for a Computer Software Configuration Item (CSCI) and the methods to be used to ensure that each requirement has been met. Requirements pertaining to the CSCI's external interfaces may be presented in the SRS or in one or more Interface Requirements Specifications (IRSs) documents referenced from the SRS.

In the following paragraphs the major required sections in the SRS document according to the 498 standard will be briefly presented. This section contains excerpts from the Software Requirements Specification Data Item Description (SRS-DID). The reader is referred to this MIL-STD-498 standard document for a complete documentation specification.

The following Table 1 shows the required sections of the SRS. The first section consists of three subsections as follows:

- The Identification subsection contains a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version (s), and release number(s);
- The System Overview contains a brief statement describing the purpose of the system and the software to which this document applies. It also must describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and support agencies; identify current and planned operating sites; and list other relevant documents
- The Document overview summarizes the purpose and contents of this document and describes any security or privacy considerations associated with its use.

The Referenced documents section lists the number, title, revision, date, and source of all documents referenced in this specification.

Section 3, the Requirements section, specifies the CSCI requirements identifying those characteristics of the CSCI that are conditions for its acceptance. CSCI requirements are software requirements generated to satisfy the system requirements allocated to this CSCI. The section includes subsections specifying the states and modes (behavioral information) as well as the capabilities (functional information) of the CSCI. The capabilities subsection is divided into several subsections describing the functional partitioning, and the detailed functional description. (Data/control flow diagrams, and state transition diagrams or equivalent, are used in developing the specification of behavioral and functional specification).

The Required states and modes subsection is necessary only when the CSCI is required to operate in more than one state or mode. A state or a mode of operations will have requirements distinct from other states or modes, this section should identify and define each state and mode. Examples of states and modes include: idle, ready, active, post-use analysis, training, degraded, emergency, backup, wartime, peacetime. The requirements for the system behavior at the idle state for example must clearly associated with that state. In general, each requirement or group of requirements in this specification must be correlated to the states and modes in which they belong. The correlation may be indicated by a table or other method in

Several examples are to be described later in this Chapter to illustrate the above specification technique. Examples can be taken from the set of case studies provided in this text or from [HATLEY&PIRBHAI 88].

3.1.2 Object-Oriented Analysis (OOA)

The OOA approach gives more attention to data specification than structured approach which gives more emphasis to functional or procedural specification. The OOA approach is centered around three general concepts: objects, classes, and inheritance. The object-oriented approach has evolved from the concepts of computer simulation which is based on simulating the activities (functions) performed on some entities (objects) of a system. Objects are created and destroyed during the simulation. Objects are the basic run-time entities in an object oriented system. Objects take up space in memory and have an associated address like a data structure. For example, The arrangements of bits in an object's allocated memory space determine the state of the object at any given moment. Associated with each object is a set of functions that define the meaningful operations on that object. Thus, an object encapsulates both state and behavior. Classes define the representation of attributes and behavior of objects. Ideally, a class is an implementation of an Abstract Data Type (ADT). An ADT consists of the following parts:

- A type name (e.g. pressures, temperatures, voltages etc.),
- An optional specification of the domain of values for the type (e.g, a 2-dimensional array of floating point values, etc.)
- A specification of allowed operations (e.g., add, multiply, inverse, transpose, etc.) defined on that type.

The implementation details of a class are private to the class. The public interface of such a class is composed of two kinds of class methods. The first kind consists of functions that return meaningful abstractions about the object instance's state. The other type of methods involves transformation procedures used to move an instance from one valid state to another. A C++ struct or class, or an Ada package can be used to implement ADTs.

Inheritance is a relation between classes that allows for the definition and implementation of one class to be based on that of other existing classes (for example the class “square matrix” can be defined based on the class “matrix”). Inheritance is the most important concept that helps us realize the goal of constructing software from reusable parts, or components rather than hand coding every system from scratch. Inheritance not only supports reuse across systems, but it also directly facilitates extensibility within a given system. Inheritance minimizes the amount of work needed when adding additional features.

The logical model, which is based on teamwork/OOA (see also [SHLAER & MELLOR 92]), consists of class diagrams, object communication diagrams, state transition diagrams (STDs), and timing diagrams. The class diagram is built using an Entity Relationship Diagram defined in the previous section. The class diagram shows the various classes and the relationship between them. The object communication diagram shows the data flow between the classes of objects in the system. Then for each class, an STD is defined showing the states of the class and the function and operation which can be activated in each state. A Data Flow Diagram is defined for each state in the STD and timing diagram is also defined for the activated operations.

3.1.1 Structured Analysis (SA)

In this section, the most widely used method for structured analysis of real-time systems is described. This method was first proposed by Ward and Mellor [Ward & Mellor 85] and then later enhanced by Hatley and Pirbhai [Hat&B 88]. It is supported by most CASE tools. The method is an extension of the structured system analysis, data flow analysis, and transaction analysis to real-time systems.

The Hatley and Pirbhai technique relies primarily on three types of graphs or models described as follows:

- The process model represented by transformation graphs which are basically Data Flow and Control Flow Diagrams (DFDs/CFDs). The model represents the processing of information in terms of data and control signals in the proposed system.
- State Transition Diagrams depict the sequence of states and the corresponding control actions and, therefore, define the control specifications which represent the real-time behavioral model of the system.
- Entity-Relationship Diagrams provide an information model for the data items and control signals and the relationships among these data items.

A fully developed specification starts with a context diagram, the highest level DFD/CFD. This graph places the software in a real-world context. It defines the whole CSCI as one component and models the interfaces to external software or hardware components that interact with the CSCI.

Other DFDs/CFDs specify the processing in the software at a lower level of detail where a component at a higher level DFD/CFD is further specified by its own DFD/CFD at a lower level. The various graphs fit together and form a hierarchy with the context diagram at the top. Nodes in these graphs are either data transformation nodes or control transformation nodes.

Processes specifications (P-specs) are used to define data transformation primitive nodes (those that do not have a lower level DFD/CFD to specify them). P-specs can be defined using pre/post conditions, structured high-level languages, or pseudo-code.

Control transformation nodes are specified further by C-specs using decision tables, State Transition Diagrams (STDs), or state-event matrices which show the control flow behind the system control processes. They specify the details of the sequence of states at which the activities (or processes) defined in the DFDs/CFDsare to take place. Decision Tables (DTs), Process Activation Tables (PATs), and State/Event Matrices (SEMs) are also used as C-specs.

The information model defined in an Entity-Relation Diagram (ERD) serves as a library of data records and describes the relationship between data elements used in the system. All the information used in the Transformation Graphs, must correspond to data derived in the ERD.

The above graphs represent three views of the system, namely, the process view (represented by DFDs/CFDs) specify the functional as well as the information flow representation, the control view (represented by STDs, DTs, PATs, or SEMs) specify the behavioral representation, and the data view (described by the ERDs) Specify the information relationship and content representation. When these graphs are complete and fully developed, they provide a complete logical specification for the CSCI.

3.1 Introduction to Requirements Analysis

Recall the description in the ESA standard for the Software Requirements Analysis (SWRA) phase presented in section 2.3.1. In the SWRA phase a logical model of the system is produced and used to analyze the completeness, consistency, and testability of the requirements. Building prototypes and dynamic simulation models in this phase could be necessary to analyze the dynamic behavior of complex real-time systems and verify and clarify their software requirements.Tool support for these tasks will be defferred to Chapter 6. A Software Requirement Document (SRD) is produced at the end of this phase to capture the software requirements specification. The SRD is formally reviewed during the Software Requirements Review (SRR) by the users, software engineers, hardware engineers, and managers concerned. The project management plan is also updated and a detailed plan for the design phase must be produced.

The logical model mentioned above is an implementation-independent model of the software. It should clearly reflect what is needed by the user and its components should be traced to requirements in the user requirements document. The user requirements document is an input to this phase and is the main source of information in building the logical model. This model is used to specify the technical software requirements. By examining the logical model a precise and detailed set of software requirements are obtained and classified as functional requirements, performance requirements, interface requirements, resource requirements, verification requirements, acceptance testing requirements, security requirements, portability requirements, reliability, quality requirements, maintainability, and safety requirements.

The specification of SWRA phase in the DOD standard MIL-STD-498 also focuses on analyzing the requirements and developing a logical model for each computer software configuration item (CSCI). The inputs to this phase is the system System/Subsystem Design Description (SSDD). This document was prepared as part of the activities outlined section 2.3.2 for the System Design (SYSD) phase. The SSDD defines and records the overall system architecture in terms of Hardware Configuration Items (HWCIs), Computer Software Configuration Items (CSCIs). It also specifies the overall appearance and behavior in response to system requirements (specified in the System requirements analysis (SYSRA) phase). Moreover, the allocation of system requirements to HWCIs, CSCIs, and manual operations is also recorded. The design information pertaining to interfaces may be found in the SSDD or in the Interface Design Descriptions (IDDs) document, and design information pertaining to databases may have also been included in the SSDD or in a Database Design Descriptions (DBDDs) document.

The documents above give the needed overall system view as well as the requirements pertaining to the specific CSCIs. The DOD standard mandates that a detailed and precise description of the functional, timing, and data requirements by means of a logical model for each CSCI must be developed and documented in this phase. A Software Requirements Specification document must be produced and reviewed.

Two major approaches for developing the logical model will be discussed next. These are the structured analysis approach (also called functional or process oriented approach), and the object oriented analysis approach. The two approaches differ in the way they model the system requirements in order to obtain precise specification. Following the discussion on the analysis techniques presented in the following subsections, a detailed description of the software requirements specifications document described in the DOD standard is presented.

Chapter III: cd groSOFTWARE REQUIREMENTS ANALYSIS

3.1 Introduction to Requirements Analysis

3.2 Structured Analysis for Real-Time Software Using ICASE

3.2.1 Introduction to Structured Analysis

3.2.2 The ICASE SA-RT tool.

3.2.3 Examples

3.2.3.1 Traffic Light (Add its requirements to Chapter 1 before AMS etc.)

3.2.3.2 ATM

3.2.3.3 AMS

3.3 Object-Oriented Analysis Using ICASE

3.3.1 Introduction to Object-Oriented Analysis and Data Modeling

3.3.2 The Current and Evolving Notations of OOA

3.3.3 The ICASE OOA tool support.

3.3.4 Examples

****

- 3.1 Introduction to Requirements Analysis
- 3.2 Structured Analysis for Real-Time Software Using ICASE
- 3.3 Object-Oriented Analysis (OOA) Using ICASE
- 3.4 Analysis using Teamwork
- 3.5 ICASE Environments: IDE's Software through Pictures
- 3.6 Examples of SART
- 3.7 References

Chapter 3 describes the methodologies for real-time software requirements analysis and specification. The Chapter starts by describing the overall conceptual modeling task, and the resulting software requirements specification document. Section 3.2 describes the structured analysis technique and the tool support given in Teamwork/SA-RT using the example requirements presented in section 1.3 of Chapter 1. The object-oriented Analysis (OOA) Methodology and notations are introduced in section 3.3.1. Several different OOA notations are discussed and contrasted with the structured analysis approach. The evolving notations for OOA such CASE/RT and the fusion method are compared with the current notation in section 3.3.2. Developing analysis diagrams using Teamwork/IM and ObjectTeam/OOA are then described using examples in sections 3.3.3 and 3.3.4.