Table 1. System engineering functions correlated to software system engineering (SwSE).

System engineering function	SwSE function	SwSE function description
Problem definition	Requirements analysis	Determine needs and constraints by analyzing system requirements allocated to software
Solution analysis	Software design	Determine ways to satisfy requirements and constraints, analyze possible solutions, and select the optimum one
Process planning	Process planning	Determine product development tasks, precedence, and potential risks to the project
Process control	Process control	Determine methods for controlling project and process, measure progress, and take corrective action where necessary
Product evaluation	Verification, validation, and testing	Evaluate final product and documentation

the effort to ensure that the project meets costs and schedules and satisfies technical requirements.[6]

Figure 3 illustrates the management relationships between project management, SwSE, and SwE. Project management has overall management responsibility for the project and the authority to commit resources. SwSE determines the technical approach, makes technical decisions, interfaces with the technical acquirer, and approves and accepts the final software product. SwE is responsible for developing the software design, coding the design, and developing software components.

THE FUNCTIONS OF SOFTWARE SYSTEM ENGINEERING

Table 1 lists the five main functions of system engineering correlated to SwSE, along with a brief general description of each SwSE function.

Requirements analysis

The first step in any software development activity is to determine and document the system-level requirements in either a system requirements specification (SRS) or a software requirements specification or both. Software requirements include capabilities that a user needs to solve a problem or achieve an objective as well as capabilities that a system or component needs to satisfy a contract, standard, or other formally imposed document.[7]

We can categorize software requirements as follows:[8]

- *Functional requirements* specify functions that a system or system component must be capable of performing.
- *Performance requirements* specify performance characteristics that a system or system component must possess, such as speed, accuracy, and frequency.
- *External interface requirements* specify hardware, software, or database elements with which a system or component must interface, or set forth constraints on formats, timing, or other factors caused by such an interface.
- *Design constraints* affect or constrain the design of a software system or software system component, for example, language requirements, physical hardware requirements, software development standards, and software quality assurance standards.
- *Quality attributes* specify the degree to which software possesses attributes that affect quality, such as correctness, reliability, maintainability, and portability.

Software requirements analysis begins after system engineering has defined the acquirer and user system requirements. Its functions include identification of all—or as many as possible—software system requirements, and its conclusion marks the established requirements baseline, sometimes called the allocated baseline.[2]

Software design

Software design is the process of selecting and documenting the most effective and efficient system elements that together will implement the software system requirements.[9] The design represents a specific, logical approach to meet the software requirements.

Software design is traditionally partitioned into two components:

- *Architectural design* is equivalent to system design, during which the developer selects the system-level structure and allocates the software requirements to the structure's components. Architectural design—sometimes called top-level design or preliminary design—typically defines and structures computer program components and data, defines the interfaces, and prepares timing and sizing estimates. It includes information such as the overall processing architecture, function allocations (but not detailed descriptions), data flows, system utilities, operating system interfaces, and storage throughput.