Editor: Richard H. Thayer ■ California State Univ., Sacramento ■ thayer@csus.edu
Technical Reviewer: Merlin Dorfman ■ Cisco Systems ■ dorfman@computer.org

# Software Design, Part 2

**design constraint:** Any requirement that affects or constrains the design of a software system or software system component (for example, physical requirements, performance requirements, software development standards, software quality assurance standards). [ANSI/IEEE Std 610.12-1990]

**design entity:** A part of a design that is structurally, functionally, or otherwise distinct from other elements or that plays a different role relative to other design entities. Each design entity is separately named and referenced. *Also called design element.*

**design fault:** A design (specification, coding) fault that results from a human error during system design and that might result in a design failure.

**design inspection:** A static analysis technique that relies on visual examination of development products to detect errors, standards violations, and other problems. *Similar to design walkthroughs.* [IEEE Std 610.12-1990]

**design language:** A standardized notation, modeling technique, or other representation scheme and its usage conventions, shown to be effective in representing and communicating design information.

**design level:** The design decomposition of the software item (for example, system, subsystem, program, or module). [IEEE Std 610.12-1990]

**design methodology:** A systematic approach to creating a design consisting of the ordered application of a specific collection of tools, techniques, and guidelines.

**design pattern:** A description of the problem and the essence of its solution to enable the solution to be reused in different settings; not a detailed specification, but a description of accumulated wisdom and experience.

**design phase:** The period of time in the software life cycle during which definitions for architecture, software components, interfaces, and data are created, documented, and verified to satisfy requirements.

**design strategies:** An overall plan and direction for performing design (for example, functional decomposition).

**design view:** A subset of design entity attributes that are specifically suited to the needs of a particular participant or stakeholder. [IEEE Std 610.12-1990]

**design-to-cost:** An approach to managing a system/software project so as to hold the project to a predetermined cost. Actual and projected costs are closely tracked, and actions such as deleting or postponing lower-priority requirements are taken if costs threaten to exceed targets. *Also called cost as an independent variable (CAIV).*

**detailed design:** 1. The process of refining and expanding software architectural designs to more detailed descriptions of the processing logic, data structures, and data definitions. This continues until the design is sufficiently complete to be implemented. 2. The result of the detailed design process. *Also called design, low-level design, module design, program design.* [ANSI/IEEE Std 610.12-1990]

**detailed design description:** A document that describes the exact detailed configuration of a computer program. It identifies the input, output, control logic, algorithms, and data structure of each individual low-level component of the software product and is the primary product of the detailed design phase. *Also called detailed design specification.*

**detailed design phase:** The software development lifecycle phase during which the detailed design process takes place, using the software system design and software architecture from the previous phase (architectural design) to produce the detailed logic for each unit such that it is ready for coding. *Also called detailed design stage.*

**detailed design review:** A milestone review to determine the acceptability of the detailed software design (as depicted in the detailed design description) to satisfy the requirements of the software requirements document.

**formal design:** The process of using a formal method for software design.

**framework:** A partially completed software subsystem that can be extended by appropriately instantiating some specific plug-ins.

**function-oriented design:** The partitioning of a design into subsystems and modules, with each one handling one or more functions. *Contrast with object-oriented design, data-structure-oriented design.*

**interface design document (IDD):** A description of the architecture and design of interfaces between system and components. These descriptions include control algorithms, protocols, data contents and formats, and performance.

**Jackson Structured Design Method (JSD):** A structured software development methodology for the analysis and design of both data-processing and real-time systems developed by Michael Jackson Systems.

0740-7459/04/$20.00 © 2004 IEEE