Systematic OO Programming with Axiomatic Design

Sung-Hee Do and Nam P. Suh, Axiomatic Design Software

Axiomatic design offers a systematic and orderly way to proceed through the software development process. The methodology ensures that developers make the best possible design decisions by providing decision-making criteria in the form of two axioms:

- The *independence axiom* suggests that the best designs maintain the independence of the functional requirements, ensuring that the design can achieve each function without inadvertently affecting any other function.
- The *information axiom* suggests that the best designs minimize their information content. Thus, the solution with the greatest likelihood of success is the simplest solution.

Software developers can

- determine the quality of the design based upon the degree to which it satisfies the defined functional requirements;
- ensure that the design satisfies each function independently and that coupling, which could cause unintended consequences, does not occur;
- enjoy easier job assignment and team management because they can use the system architecture diagram to assign tasks and define the interrelationships between various modules or classes; and
- handle software change orders quickly and easily because the system architecture identifies the modules affected by a change.

Editors: Jerzy W. Rozenblit, University of Arizona, ECE 320E, Tucson, AZ 85721; jr@ece.arizona.edu; and Sanjaya Kumar, Honeywell Technology Center, MS MN65-2200, 3660 Technology Dr., Minneapolis, MN 55418; skumar@htc.honeywell.com

The Acclaro design software's development offers a case study of axiomatic design's effectiveness.

Axiomatic design lets the developer document the design fully, making it easier to change or add extensions.

HOW IT WORKS

Chronologically, axiomatic design begins with a thorough investigation of customer attributes to discover customers' needs. Developers note these needs, then determine the design's functional requirements (FRs) by answering the question, "What must this design do to satisfy our customers' needs?" Developers determine the answers and record the results as FRs in the left column of a design matrix.

Across the top of the matrix, developers record the design parameters (DPs)—physical solutions that satisfy each functional requirement. More than one design parameter may satisfy a particular function, but the information axiom will help determine the best solution. Developers may indicate a relationship between the functional requirement and the design parameter with an X in the matrix or with an equation. The matrix lets the developer check that each design parameter satisfies only one functional requirement. Developers decompose the design, working through each part to determine the subfunctional requirements and subdesign parameters of their previous design decisions until they've completed the entire design.

The V-shaped diagram in Figure 1 shows the axiomatic development process for software. Some software development tools begin at the bottom of the V with module definition, giving little attention to top-down conceptual design. Axiomatic development pushes the development team to define the problem explicitly, envisioning the entire design before writing the first line of code.

ACCLARO CASE STUDY

The Acclaro design software is the first implementation of a software package using axiomatic design principles. We developed it as an axiomatic design tool that will help developers apply axiomatic design techniques to their software projects. Our software's development provides a case study in the benefits of applying axiomatic design to object-oriented programming. Testing the Acclaro software took only 10 percent of the software's development time—one measure of this project's success. Axiomatic design

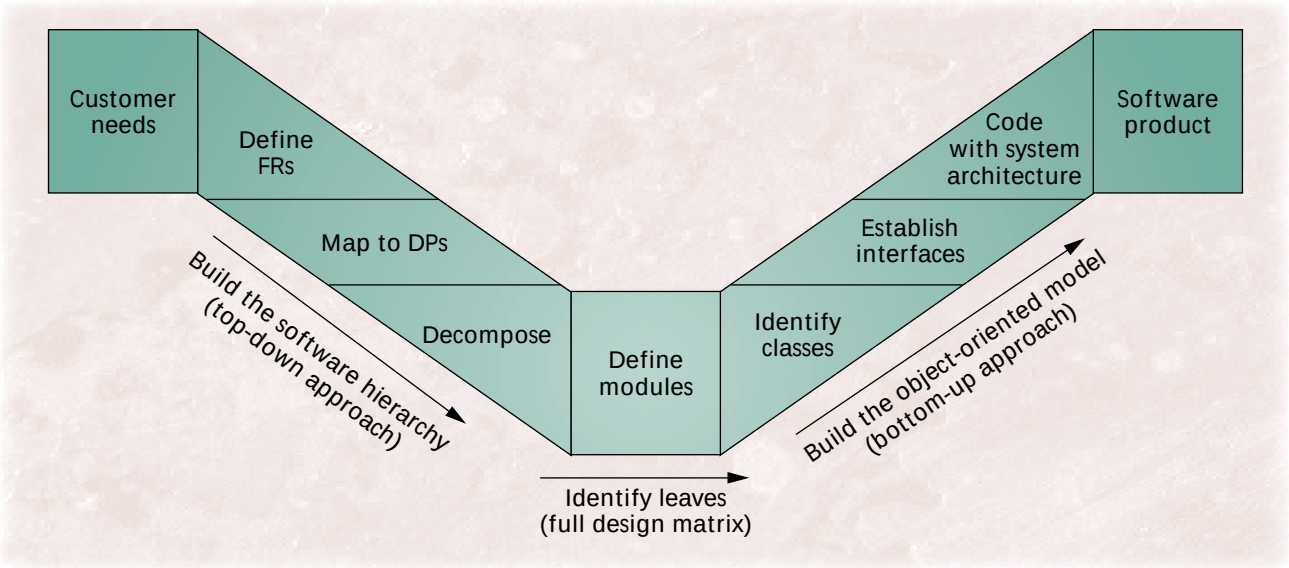

Figure 1. Axiomatic design uses a top-down approach to create software. (Diagram adapted from a 12 May 1999 Society of Automotive Engineers' workshop presentation on Reliability, Maintainability, Supportability, and Logistics by Basem El-Haik, Ford Motor Company.)

also let us seamlessly combine the code generated by external developers who worked in parallel with the main development team.

Acclaro guides the developer through the axiomatic design process, documenting the design, decisions, and rationale. The software assists in the design analysis and offers recommendations for creating the best possible designs. Acclaro also links to existing development tools, aids communication between development team members, and provides information that can make project management easier. Acclaro is a multiuser, platform-independent application that captures extensive design details and performs complex matrix manipulations.

Using axiomatic design methods, Acclaro's developers began the design process by stating their task in terms of what the software must accomplish at the highest level. Acclaro's highest level functional requirement was "to help developers succeed with software design." As the developers describe each functional requirement, they must find design parameters to satisfy that requirement, then choose the one DP most likely to provide a feasible solution. In the case of Acclaro, the highest level DP was "create axiomatic design software."

Developers must also determine if there are any constraints on the functional requirement that they must consider in the decision process. For Acclaro, a high-level constraint was that the software remain "independent of the operating system" so that it was portable.

Developers then defined the next level of functional requirements and selected associated DPs. In Table 1, the P in the leftmost column represents the parent level; the items in the numbered rows below it represent Acclaro's highest level sub-FRs and sub-DPs.

Table 1. First-level decomposition of Acclaro.

X	Functional requirements FR1.*x*	Design parameters DP1.*x*
P	Help designers succeed with software design	Create axiomatic design software
1	Manage design workflow	Design road map
2	Provide decision-making environment	Decision-making criterion
3	Support software's ease of use	Graphical user interface (GUI)
4	Provide efficient data input/output	Data manager
5	Provide utility function	Plug-in software

The developers continue this process from the system level down to individual modules in increasingly specific detail. They describe the functional requirements and associated DPs for the entire software system until they decompose the design into manageable pieces.

The design matrix lets developers track this decomposition, helping to ensure that design parameters satisfy the functional requirements and are not coupled. Figure 2 shows a portion of the design matrix from a fourth-level branch of Acclaro.

Acclaro defines software modules by function. Developers can precisely define modules as FR leaves and can ensure that each module performs its function without unintentionally affecting other functions.

Derived from the design matrix, the system architecture tells developers the sequence for creating operations and modules in the software. Figure 3 shows the corresponding system architecture section for the design matrix section shown in Figure 2. The Xs in the matrix section, which represent specific functions, determine the system architecture's form. The topmost FRs and DPs are called parents. Axiomatic design assigns a number to the first level that further defines the parent level. A sub-FR or sub-DP of this level acquires an additional number. For example, FR11, FR12, and FR13 are all sub-FRs of FR1.

						DP1141								
						2								
						1	2	3						
								4				1	2	3
								1	2	3	4			
						Attribute for FR	Attribute for GUI	Copy method	Cut method	Paste method	Drag-and-drop method	New method	Change method	Delete method	Module information
FR1141	2	1			Describe the FRs	X									M114121
		2			Display to the graphics		X								M114122
		3	4	1	Support copying	X	X	X							M11412341
				2	Support cutting	X	X		X						M11412342
				3	Support pasting	X	X			X					M11412343
				4	Support moving	X	X		X	X	X				M11412344
			1		Make a new storage	X	X	X	X	X	X	X			M1141231
			2		Support changing	X	X	X	X	X	X		X		M1141232
			3		Support deleting	X	X	X	X	X	X			X	M1141233

Figure 2. Fourth-level branch of the Acclaro design matrix.

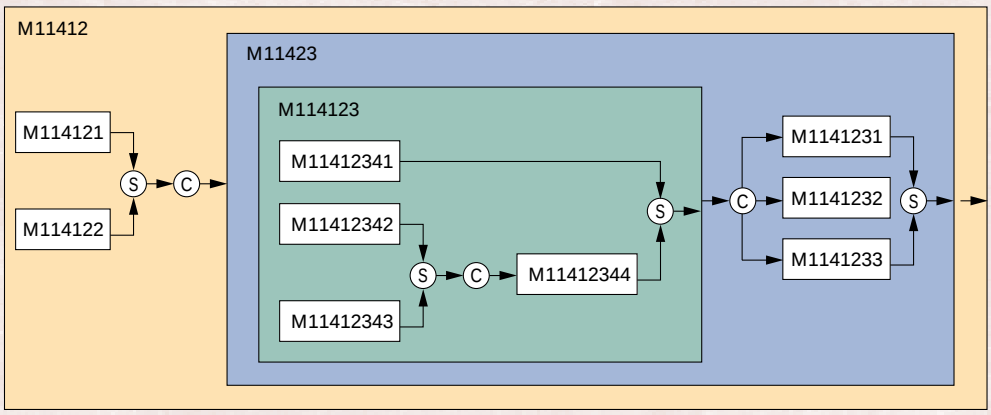

Figure 3. Corresponding system architecture for the design matrix in Figure 2. The circled Ss represent summations of modules; the circled Cs represent control points that indicate one step must precede another.

The design matrix excerpt shown in Figure 2 falls within module 1141 of the system architecture. The outermost layer of the matrix is 1141, the next deeper layer is 11412. Module 1141 has a submodule, 11411, and possibly other submodules not shown in this cross section. Module 11412 is five layers deep from the topmost layer.

Within the next layer are modules 114121, 114122, and 114123. Developers can make decisions for modules 114121 and 114122 in parallel because these two modules are independent of one another—only diagonally positioned Xs in the matrix describe the relationships between these FR-DP pairs. (Red lines border each FR-DP pair in Figure 2.) This diagonal relationship means that each design parameter is independent and affects only the functional requirement it is matched to satisfy.

Module 114123, however, is decoupled because it is impacted by modules 114121 and 114122, as captured by the off-diagonal terms that fall under DP114121 and DP114122. As long as module 114123 follows modules 114121 and 114122, it compensates for the impact of the off-diagonal terms on 114123. Following

these rules, developers must make decisions about module 114123 only after making those for modules 114121 and 114122. Thus the system architecture serves as a graphical representation of the matrix: It gives the order in which to make design decisions and indicates how those decisions will impact other design functions.

The parts of the software system represented by the matrix's diagonal terms are the software's core functions and should be developed in order, from left to right down the diagonal. Off-diagonal terms represent other relationships that developers should consider in developing software, but they do not represent core functions. The matrix helps the developers establish and organize these relationships.

DP leaves can each represent attributes, methods, or arguments of a specific class. One or two levels up, the DPs can also be defined as a class combining the leaves below, depending on how the developer has broken down the design. Once developers define the classes, they can define their attributes and operations. Next, they can create interfaces by establishing relationships between objects and operations, using the design matrices created earlier.

Our development of Acclaro showed us that axiomatic design is a powerful tool for creating new designs. Developers can also use this methodology to decompose existing designs, create a system architecture drawing, and diagnose problems. We've already begun work on a new version of Acclaro with features geared specifically to software developers.❖

Sung-Hee Do is vice president of technology for Axiomatic. Contact him at dosh@axiod.com.

Nam P. Suh is head of the department of mechanical engineering and director of the Manufacturing Institute at MIT. He is also an Axiomatic board member. Contact him at npsuh@mit.edu.