A TAXONOMY OF DECOMPOSITION STRATEGIES BASED ON STRUCTURES, BEHAVIORS, AND GOALS

Philip J. Koopman, Jr.
United Technologies Research Center
East Hartford, Connecticut

ABSTRACT

Designs are often decomposed into subdesigns in a divide-and-conquer approach to dealing with complexity. A wide variety of strategies are in common use for accomplishing decomposition. This paper describes a taxonomy of decomposition strategies based on the design attributes of structures, behaviors, and goals as an aid to understanding when each different strategy may be appropriate. The taxonomy is descriptive rather than prescriptive; it concentrates on providing a common ground for representing both business-driven and technically-driven design decisions. An example is presented that suggests hierarchically layering diverse decomposition strategies can accommodate diverse designs and design processes.

INTRODUCTION

When confronted with creating a complex system, designers rely on decomposition strategies to break things into manageable chunks. Typically, multiple layers of successively finer-grain decompositions are employed to break the design into smaller and smaller pieces until individual design elements are tractable. The experience of the design team, precedents from previous designs, availability of pre-designed components, capabilities of design automation tools, and the maturity of artifact and process technology all influence the degree to which a design must be decomposed.

While decomposition is widely employed as a problem-solving approach, there are significant variations in the criteria used for performing decompositions in practice. Not all of these criteria are based on a strictly technical point of view — in fact, some of the strategies used every day in industry have little to do with technology. But, these non-technical strategies have historically served a useful purpose in interfacing technical design activities with the non-technical world of turning a profit in business. It seems advantageous to encompass business and management considerations as they affect practical design theory work. Thus, this paper presents a framework that provides a common ground for representing both technical and non-technical design decomposition decisions.

Rather than inventing new strategies for decomposing designs, this paper instead creates a taxonomy that helps describe certain strengths and weaknesses observed to be in common with strategies that fall into a each category. By grouping strategies, it provides generalizations that could in the future be used as the basis for selection guidelines in a methodical design approach.

The diversity of techniques successfully employed in industrial practice suggests that there is no single "best" decomposition strategy. Rather, using diverse strategies seems to be necessary whenever the design task must reconcile technical considerations with non-technical design criteria. This is in contrast to the usual prescriptive design methodology view of applying a uniform, technically-based decomposition approach (*e.g.,* conceptual design with a "function structure" in Pahl & Beitz, 1984).

A methodology to select decomposition strategies is left for future work, and will probably have to include technical, organizational, and business considerations. This paper presents a potentially unifying view of the options available as a necessary first step in such a methodology.

DECOMPOSITION BY STRUCTURES, BEHAVIORS, AND GOALS

While the content of a design can vary widely depending on technical domain and level of abstraction, it is common to characterize a design as having elements that fall into a

small number of attribute categories. Often, the pair of attribute categories "form" and "function" are used, although terminology varies. For the current discussion, it is advantageous to use three attribute categories called *structures* (a variation of "form"), *behaviors* (a variation of "function"), and *goals* (the desired emergent properties).

Goals are included as a third category of the design representation itself rather than an external criteria against which the results are measured. The initial reason for doing this was that design practice often treats design goals on a par with structures and behaviors in terms of tradeoffs (*e.g.,* a failure to meet cost goals can be resolved by redesigning a structure, but may also be resolved by raising the cost target if the designers are pressed for time).

Recently, Chi *et al.* (1994) have suggested that "matter", "processes", and "mental states" form the three primary ontologies for reasoning and learning about scientific concepts (these categories appear to be substantially the same as structure, behavior, and goals). They claim that conceptual shifts within categories are much easier than conceptual changes between categories. In order to support human design activities, it seems advantageous to use a representational framework that is compatible with the way people think. Thus, including goals as a category may help represent designs in terms similar to the way engineers think about technology.

The following sections describe more fully what is meant by structures, behaviors and goals. While concise definitions are given, the reader is encouraged to use the examples to help tailor these terms for specific design situations.

Structures

Structures are physical components, logical objects, geometric attributes, fields, or arrangements of other structures within a design. Structures typically answer the question of "what" in a design, and typically are described using nouns and adjectives.

The term structure is used to encompass not only geometry and other tangible aspects of "form", but also non-tangible entities (*e.g.,* magnetic fields, data representations), material composition, and business organization.

Examples of structures from various disciplines include:

- Mechanical artifacts: gears, linkages, beams, assemblies, geometric shape
- Chemical processes: floorplan, reactor, pump, plumbing topology
- Fluids: turbine blades, venturi
- Digital hardware: circuit board, chip, register, gate, integrated circuit layout
- Software: object, database, data element, control variable
- Electronics: resistor, capacitor, wire, magnetic field
- Business: the Engineering Department, the Sales Department

Behaviors

A behavior is an action, force, process, or control law that is exerted on or by a structure with respect to the structure's external environment. In the case that only a portion of a design (a *subdesign*) is under consideration, other subdesigns constitute a portion of the external environment for the behavior under consideration. Behaviors typically answer the questions of "how" and "when" in a design, and are typically described using verbs and adverbs.

Behaviors encompass not only data transformation and causal relationships normally associated with the word "function", but also processes, flows, and other temporal aspects of the design. In the computer field, functions often specifically exclude timing and sequencing aspects of the design; using the term behavior should help avoid confusion in that domain.

Examples of behaviors from various disciplines include:

- Mechanical artifacts: coupling the endpoints of a linkage, supporting a load, thrust
- Chemical processes: containing spills, combustion
- Fluids: changes to pressure, temperature, and flow speed
- Digital hardware: execution of software, retention of data
- Software: transformation of data, execution of an algorithm or method
- Electronics: voltage regulation, storage of energy potential, current flow
- Business: execution of a business process such as order fulfillment

Goals

Goals are emergent design properties that satisfy the needs which the design is intended to fulfill. Goals include any result that is not directly available as an "off-the-shelf" building block. Goals thus include performance targets, costs and aesthetics. Goals also include emergent structures and behaviors that are too complex to be implemented without some sort of decomposition (*i.e.*, a design goal might be to provide a certain emergent structure such as a specialized chair, but that result must be produced by decomposing the design into structures of legs, back, and seat).

Clearly, classification of a design attribute as a goal depends on the state of the art in the particular technology of interest as well as the design context. In particular, the goal for a component will often be to provide a structure or behavior included in a higher-level design (*e.g.,* the goal of a linkage design is to couple specified end-point trajectories, whereas a design incorporating that linkage specifies a generic linkage structure and particular trajectory behaviors without considering how that linkage is actually implemented). Goals typically answer the question "why" when trying to explain decisions made with respect to structures and behaviors.

Examples of goals from various disciplines include:

- Mechanical artifacts: maximum size and weight, minimum operating cycles before repair, cost
- Chemical processes: production of pollutants, conversion efficiency, equipment cost
- Fluids: weight, turbulence, noise, efficiency, cost
- Digital hardware: size limitations, computational throughput, power dissipation, reliability, cost

- Software: program size limitations, efficiency in use of hardware, limits to number of "bugs", cost
- Electronics: density, electromagnetic radiation, performance under varying environmental conditions, cost
- Business: customer satisfaction, profitability

External constraints due to technological, business, regulatory, or political considerations are beyond the scope of this discussion. In a method that uses the descriptive techniques presented here, constraints will appear as a combination of goals and limitations on acceptable decompositions.

Decomposition Strategies

A *design decomposition* consists of a design and its constituent subdesigns. When the "child" subdesigns are combined, the emergent result satisfies the "parent" design's stated behaviors, structures, and goals. A *decomposition strategy* is a method for selecting subsets of attributes to be assigned to child subdesigns.

Figure 1 shows an ad-hoc decomposition in which a design has structures $S_1, S_2, .., S_n$; behaviors $B_1, B_2, .., B_p$; and goals $G_1, G_2, .., G_q$. The multiple subdesigns *1* through *m* resulting from decomposition contain potentially modified versions of the original structure, behavior, and goal attributes. For example, if G_1 is a weight goal, $G_1{}^1$ through $G_1{}^m$ would be the weight goals for the subdesigns *1* through *m*. Similarly, S_1 might be a structure which is actually an assembly of components $S_1{}^1$ through $S_1{}^m$, and B_1 might be a behavior which emerges from an interaction of behaviors $B_1{}^1$ through $B_1{}^m$.

While an ad-hoc decomposition may provide a globally optimal result (with respect to the design being decomposed), such an approach seems to yield no insight. It is, in effect, a decomposition strategy based on art, not science. While an ad-hoc design may be acceptable for long-lived designs that evolve slowly over time, that approach may well fail for short-lived designs that must be brought to market rapidly or changed regularly. In contrast to an ad-hoc approach, methodical design decompositions are characterized by separation of attributes among subdesigns rather than a distribution or dilution of attributes among subdesigns.

Figure 2 shows a structural design decomposition, in which structures S_1 through S_n are allocated uniquely to *n* different subdesigns. (It is possible that sets of multiple structures could be allocated to particular subdesigns, but this discussion considers such multiple structure sets to be subsumed under a single umbrella structure.)

In this structural decomposition, it is now clear that the subdesigns are uniquely assigned different structural portions. While potentially a full range of behaviors and goals might still be associated with each subdesign, the superscripts indicate that usually some adjustments will be made as in the preceding ad-hoc example. Decomposition based on behaviors or goals would be represented similarly, with individual behaviors or goals uniquely allocated to subdesigns respectively.

Decomposition could also be performed using combinations of structures, behaviors, and goals. Figure 3 shows behavior and structure pairs used as the basis for decomposition, which is denoted as a structure+behavior decomposition. In this example, one structure+behavior pair is assigned to each subdesign.

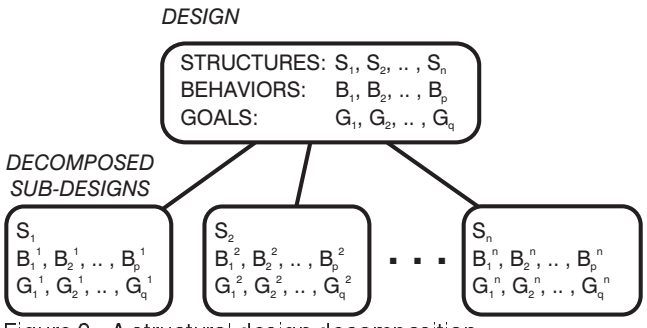

Figure 2. A structural design decomposition.

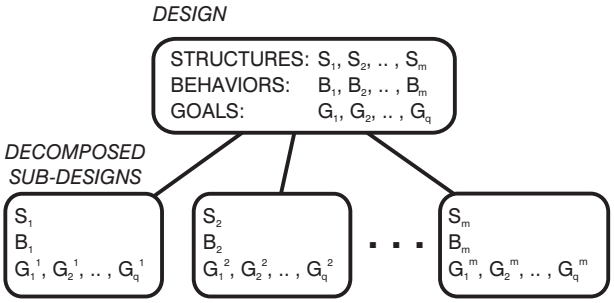

Figure 3. A structure+behavior combination design decomposition.

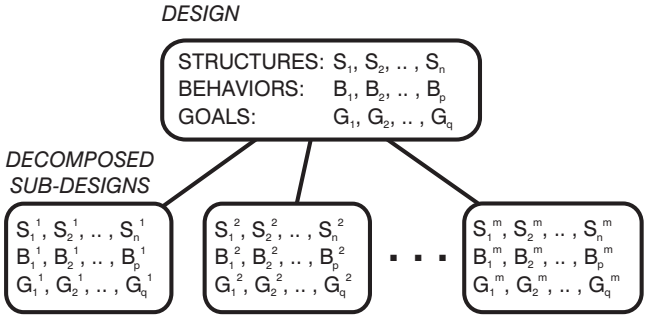

Figure 1. An ad-hoc design decomposition.

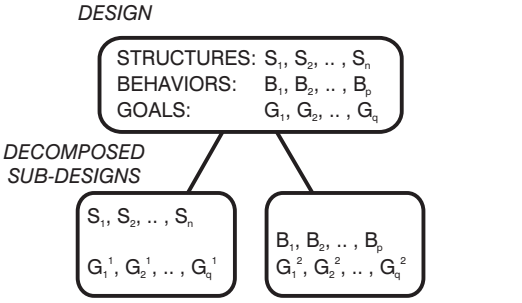

Figure 4. A structure/behavior split design decomposition.

A further possibility is decomposition by splitting attributes, as in a structure/behavior split shown in Figure 4. When performing splitting, two attribute types are decoupled so as to be largely independent, and thus are assigned to separate subdesigns. In the example of Figure 4, one subdesign is responsible for providing structures, while a separate subdesign is responsible for providing behaviors.

Figures 3 and 4 show only *complete* decompositions, in which all subdesigns are encompassed by the appropriate separation of attributes. *Partial* decompositions are also possible, in which only a portion of the design is decomposed according to a particular strategy, with the remainder of the design lumped into a single subdesign for further decomposition using some different strategy. Additionally, the figures show *homogeneous* decompositions, in which only a single strategy is used. It is possible to employ *heterogeneous* strategies, in which one portion of a design is decomposed with one strategy, and the remainder of the design is decomposed using a second strategy.

A TAXONOMY OF DECOMPOSITION STRATEGIES

Figure 5 shows a taxonomy of decomposition strategies. The italicized examples are a sampling of actual strategies encountered in industry and the literature that fall into each category. The arcs between strategies indicate that split and combined strategies both have strengths and weaknesses related to those of neighboring pure strategies.

Generalizations about different decomposition strategy types can provide guidance as to when methodologies based on those strategies are appropriate to use. While a rigorous theory does not yet exist, a first step is to create a list of *pros* and *cons* for each strategy based on industrial experience as well as example situations in which such strategies are most often employed in practice.

A common theme in this discussion is a fundamental tension between decoupling of subdesigns and cross-subdesign optimization. Decoupling is desirable because it reduces the complexity of a design by limiting the number of structures, behaviors, and goals that must be considered at one time. Additionally, decoupling makes it simpler to hold design teams accountable for satisfying all subdesign requirements. On the other hand, decoupling limits opportunities for cross-subdesign optimization. In decomposition strategies that facilitate optimization, the tradeoff seems to be that accountability for providing individual subdesigns is compromised by an increased coupling among design decisions.

"Pure" Decompositions

"Pure" decompositions, that decompose by a single attribute type, have substantial drawbacks because they leave two of the three attribute types spread throughout the resultant subdesigns. This spreading out of attribute types can lead to complex coordination problems in creating the subdesigns and later subdesign integrating efforts. For example in a structural decomposition no particular structure (and, by extension, potentially no particular design team) is responsible for providing any given behavior — the behavior is expected to emerge from the interaction of structures. On the other hand, those same two attribute

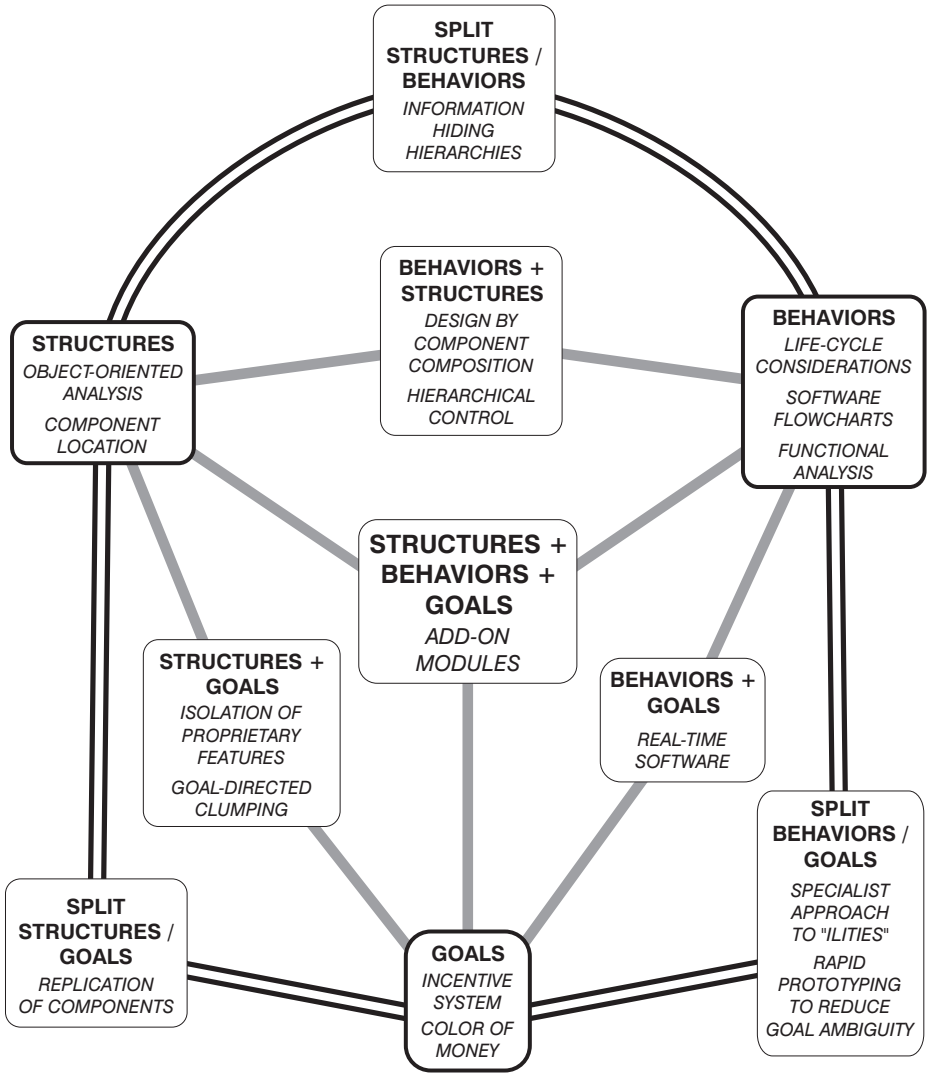

Figure 5. A taxonomy of decompositions based on structures, behaviors, and goals.

types that are spread through the subdesigns can be subject to global optimization, with no constraints on how they are allocated or shared. To continue the structural decomposition example, it may be that a particularly clever scheme for emergent behaviors may be more efficient to implement than fixing specific behaviors to specific structures — selecting a decomposition strategy that does not specify behavioral allocation leaves flexibility to achieve global optimization within the design being decomposed.

Structural Decomposition. Structural decomposition is exemplified by object-oriented analysis (*e.g.,* Rumbaugh *et al.* 1991) as well as decomposing designs based on component location (*e.g.,* the body of a container *vs.* its lid). Pahl & Beitz (1984) use a structural decomposition in the embodiment design phase when they consider the concept of function carriers.

- *Pro:* Can be used to readily subdivide both design and supplier responsibilities for manufactured items (each supplier is responsible for providing a component). This facilitates using multiple external suppliers as well as employing geographically or culturally separated internal design teams.
- *Pro:* Different fabrication technologies can be assigned to different structures, potentially reducing the need for close interaction and coordination among multiple engineering and manufacturing organizations.
- *Con:* Diffuses responsibility of ensuring behavioral correctness and meeting goals.

Behavioral Decomposition. Functional analysis (*e.g.,* Blanchard & Fabrycky 1989, Hemenway 1989) is described as organizing system requirements by desired behaviors as an early step in system engineering. The function structure of Pahl & Beitz (1984) is a similar approach for conceptual design. Life cycle considerations (*e.g.,* portions of draft MIL-STD-499B, 1992), also from the system engineering domain, can be considered a behavioral decomposition when used to subdivide business or government organizations (*e.g.,* the creation of a design team, manufacturing team, support/logistics team, and decommissioning team). In the software domain, traditional flow charting techniques lead to software organized along the lines of the flowchart boxes, where each box is a behavioral element.

- *Pro:* Can highlight temporal separation of different aspects of design life cycle, and thus emphasize that issues such as disposal must be addressed.
- *Con:* Current software engineering thinking (based on object-oriented techniques) is that a purely behavioral approach is inappropriate for complex system design.
- *Con:* Diffuses responsibility for ensuring that structures are feasible and that goals are met.

Decomposition by Goals. Decomposition by Goals is typically driven by organizational, not technical reasons. One practice is decomposition by "color of money", in which different reservoirs of money are allocated for different purposes. For example, preset budgets and staffing may force division of subdesigns according to design staff technological expertise (*e.g.,* if programmers are in short supply, more of a computer-based system will be built in hardware). In another example, corporate or manufacturing infrastructure may be designed using component parts rather than bought whole in order to evade capital equipment procurement procedures and associated delays.

A further example is that some designs may receive favorable tax treatment. Thus, there may be significant incentive to skew designs of business organizations, research programs, and technical systems (*e.g.,* requiring the ability to use alternative fuels) because of these rules.

- *Pro:* Concentrates effort on improving performance with respect to goals. For example, a prominently displayed chart might reflect progress in reducing overhead spending rates.
- *Con:* Excessive focus on goals can promote "gaming" on the part of designers. Loopholes in the goal statements may be exploited, and it is possible to create designs that meet all stated goals, but fail on the implicit goal of "does it work?"
- *Con:* Goals must be set very carefully, because the true implications may not be understood when indirect but measurable quantities are used rather than the actual, but "soft," desired outcomes. An oft-repeated aphorism is "be careful what you ask for — you might get it."

Split Decomposition Strategies

In many cases it is desirable to reduce coupling resulting from the attribute spreading encountered in pure strategies. This can be accomplished by splitting attribute types to separate them into different subdesigns, then using pure decomposition strategies on those resultant subdesigns. Split decompositions ultimately lead to the pros and cons applicable to the underlying pure strategies.

Split Structure/Behavior Decomposition. Splitting structure and behavior forms the classical approach to designs incorporating embedded digital computers, in which the computational structure is addressed by hardware engineers, and the system behavior is the responsibility of software engineers. This is not to say that, for example, the software has no structure; rather, it means that the behavior of the embedded computer is provided by the software, and that the underlying structure of the software itself is largely irrelevant as long as the correct behavior is provided. That the software has both structure and behavior is considered in decompositions performed by the software engineers.

One design methodology that exploits this type of split decomposition is information hiding hierarchies (*e.g.,* Britton & Parnas 1981).

- *Pro:* Permits changes to the behavior for increasing product functionality without changing the structure. For example, software updates can add features to embedded computer systems.
- *Pro:* Changes to structures can be accomplished without modifying behavior. For example, a sheet metal structure might be made thicker as part of an inventory

rationalization program as long as the thickness of the sheet metal does not affect design behavior.
- *Con:* Discourages optimization involving structure/behavior tradeoffs at lower levels of decomposition.
- *Con:* In practice, it is difficult to assess effects of decoupled design decisions on performance.

Split Structure/Goal Decomposition. Splitting structures and goals can be used to achieve various levels of performance by replicating structures. An example is replicating identical structural members to provide varying levels of performance (*e.g.,* load-bearing capability) or reliability (especially in the area of fault tolerant computing using modular redundancy).
- *Pro:* Various levels of performance can be provided by replicating a common structure.
- *Con:* Structures must be carefully designed to be *scalable*. In the mechanical and biological domains, the fact that volume increases more rapidly than surface area for many structures presents a limit to scalability (*e.g.,* thermal transfer, ability to employ an exoskeleton). Amdahl's Law (Amdahl 1967), which points out that adding replicated resources is vulnerable to even the slightest performance bottleneck, is a part of computer folklore that predicts true scalability can be very hard to achieve.

Split Behavior/Goal Decomposition. Splitting behaviors from goals can help meet goals that are not addressed by available design methods and support technology. For example, in technologies where integrated support for "ilities" (*e.g.,* maintainability, manufacturability) has been historically unavailable, specific designers have been tasked with taking corresponding distinct "views" of the system. These split-out goals are then considered in the rest of the decomposition only in the form of external constraints (the expert tells the designers that they have an unacceptable design) rather than in the form of explicit goals.

Another case in which behavior/goal splits occur is when some goals are unknown by the design team because of such factors as an inability to articulate behavioral requirements ("I'll know it when I see it"), political agenda, or military secrecy. In some contexts (*e.g.,* architecture) similar causes may provoke a structure/goal split instead. It seems likely that split behavior/goal decomposition would be employed in a partial or heterogeneous decomposition, with only some goals split from the rest of the design.
- *Pro:* Accommodates situations in which designers have no objective way to measure achievement of goals (*e.g.*, rapid prototyping in order to clarify goals).
- *Pro:* Accommodates business strategy requirements to have a "goal owner" who maintains tight control.
- *Con:* This approach is often not in the spirit of fully methodical design, in which subdesigns have complete internal representation of all relevant information.
- *Con:* Can lead to wasteful design iterations with no assurance of convergence.

Combined Decomposition Strategies

The major problem with the decomposition strategies examined thus far is that they eventually lead to pure decompositions which favor global optimization at the expense of increased subdesign coupling. In many designs it is advantageous to combine two attributes as the basis for decomposition in order to promote decoupling of subdesigns.

Combined Behavior+Structure Decomposition. In combined behavior+structure decomposition, each subdesign is formed with a coupled behavior/structure attribute pair. A suitably restricted set of goals is incorporated into each subdesign.

Hierarchical control (*e.g.,* Albus 1992), among many control strategies, has at the lowest levels of decomposition a tuple of sensors and actuators (structures) that are coordinated to control some system. The control loops (behaviors) are associated with each group of structures, with higher levels providing supervisory control for multiple control loops.

Design methods that advocate generating trial solutions and then performing evaluation (*e.g.,* conceptual design in Pugh, 1990) can result in creating structure+behavior subdesigns that are then evaluated against goals at the system design level.

The classical strategy of combining components from a catalog also fits into this category. When using pre-manufactured components, structure and behavior are predetermined, and the design method is to select appropriate combinations of components to produce the desired result. This example points out that a bottom-up approach can contribute to creating a design that has a rigorous top-down decomposition.
- *Pro:* Creates a clean partitioning of design responsibilities, except for meeting goals. This is typically handled by estimating budgets for contributions to system goals for each subdesign when performing the partitioning.
- *Pro:* Encapsulates implementation decisions within subdesigns.
- *Con:* Requires mediation between subsystems for goal satisfaction. Lack of appropriate mediation can result in unduly constrained design requirements for some subsystems, and ultimate failure to develop a working design.
- *Con:* May require redundancy in subdesigns (prohibits sharing of behaviors or structures), thus increasing cost. For example, multiple subsystems may each have a CPU to provide computational power.
- *Con:* Requires extreme care in defining and maintaining subdesign interfaces, because subdesign owners expect to exploit their isolation from other subdesign implementation decisions. Historically, it is not possible to get such interfaces 100% complete and accurate. This has been a great source of difficulty in performing system integration of the completed subdesigns in military systems.

Combined Structure+Goal Decomposition. Combining structures with goals in a decomposition simplifies

ensuring that goals are reached in some situations. Goal-directed clumping (*e.g.,* Marks *et al.* 1993) is an example that can be used to separate different categories of recyclable materials from non-recyclable materials as well as to meet other goals.

Suh (1990) employs a variation on this theme by employing concurrent pure goal and pure structure decompositions that are reconciled at each layer of the design (thus forming a composite decomposition combining structures and goals).

- *Pro:* Permits optimization for constrained operating environments (*e.g.,* weight, size, power consumption, operating temperatures). In particular, permits separating subdesigns that must exist in extreme environments from other portions of the system.
- *Pro:* Permits separation of subdesigns with respect to business strategies (*e.g.,* proprietary/trade secret status, capacity management for design and fabrication processes, and targeting subdesigns to known strengths of subdesign vendors).
- *Con:* It may be difficult to coax correct behavior from the resultant system. As an example, this is a traditional embedded computer system approach that ignores software issues until near the end of the design cycle, often with disastrous results.

Combined Behavior+Goal Decomposition.

Combining behavior+goal pairs when decomposing a design permits addressing goals with specific behaviors. An example from the embedded computer world is organizing software according to multiple tasks that must operate within stated deadlines, typically under the control of a real time operating system. In such designs, the goal is to produce various behavioral responses to stimuli within the goals of associated time deadlines (the deadline is a goal rather than a behavior because direct synthesis of any particular desired software execution speed is quite difficult).

- *Pro:* Where applicable, reduces design interactions caused by multiple goals (and thus isolates against goal changes) by assigning a distinct behavior to accomplish each goal.
- *Pro:* Permits dealing with time-critical behavior (deadlines as a design goal).
- *Con:* Requires coordination between subdesigns for shared structures, including competition for fixed resources.
- *Con:* Can obscure opportunities to share behaviors between overlapping goals.

Combined Structure+Behavior+Goal Decomposition.

It may be possible to decompose by simultaneously coupling structures, behaviors, and goals when creating subdesigns. An example of such a decomposition is an add-on module that provides additional functionality to an existing design (*e.g.,* an add-on diagnostics computer with sensors that is externally attached to a motor).

- *Pro:* Allows a cost/benefit tradeoff by modularly satisfying an additional goal with a stand-alone subdesign. This may form the essence of a business plan for aftermarket vendors of add-on components.
- *Pro:* Forms an evolutionary path for upgrading existing products, especially when the base product is an optimized, complex design that is difficult to change.
- *Con:* The coupled base design plus add-on module are likely to be less efficient and more expensive than an integrated design. The magnitude of this drawback depends on the relative production volumes of the original design *vs.* add-on modules as well as the business plan (*e.g.,* modernization *vs.* new equipment sales).
- *Con:* In order to be truly independent, an add-on module must exploit whatever attachment opportunities may happen to exist with other subdesigns. If changes occur to the other subdesigns, there are no explicit design linkages that ensure the add-on module will continue to meet its goal.
- *Con:* Interactions between independent add-on modules may lead to undesirable results or even system failure (*e.g.*, interaction between an active vibration control module and a vibration-monitoring diagnostics module).

Ad-hoc Decomposition

In ad-hoc decomposition, structures, behaviors, and goals are all spread across subdesigns. It is not clear how such a decomposition could be amenable to rigorous treatment. Nonetheless, there are several reasons beyond the trivial case of poor designs that such decompositions do exist.

Ad-hoc decompositions may be used to convert a difficult design problem into one or more simpler approximations to the original design problem using an intuitive leap. The strategy is to take the original design and convert it into some other similar design that will be "close enough" to meeting the real customer goals, as opposed to the articulated customer goals. This means that the decomposition is operating in the problem space, rather than the solution space. To the extent that the resultant subdesigns are in fact different design problems, this sort of transformation is beyond the scope of this taxonomy. To the extent that such ad-hoc approaches are simply short-cuts, such decompositions should be able to be represented as a series of decompositions of various categories within this taxonomy. How to do this remains an open research question.

A second use for ad-hoc decompositions is for highly optimized design. When designs have been greatly optimized, especially for performance, it may be that all traces of methodical approaches have been obscured. The prototypical example of a messy but "optimal" system is hand-tuned assembly language software that has been squeezed into a memory half the size that would be required when compiling from a high level language.

- *Pro:* A free hand in trading off all attributes among subdesigns allows the greatest possible optimization.
- *Con:* Ad-hoc optimized designs can be "brittle" by way of being difficult to modify and support.
- *Con:* Failing to follow a clear decomposition strategy may result in a lack of complexity reduction in the subdesigns

(the number of potential interactions between attributes may not be reduced).

Because ad-hoc decomposition can fail to reduce design complexity, it does not necessarily support the primary purpose of performing a decomposition in the first place. On the other hand, optimality or near-optimality is often a valuable outcome.

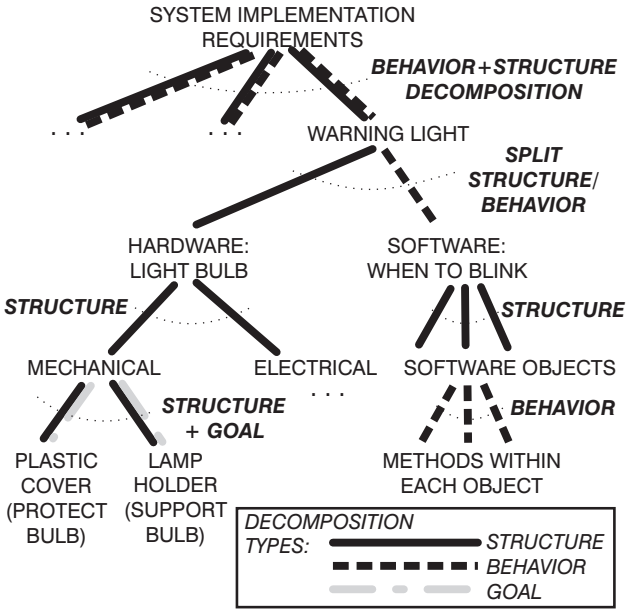

Figure 6. An example multi-level design decomposition.

EXAMPLE

Figure 6 shows an informal diagram of an example set of decompositions. The system being designed has an implementation requirement (behavior+structure decomposition) to blink a warning light when the system is not operating properly. A split structure/behavior decomposition results in hardware (structure) for the light and software (behavior) to control when the light blinks.

The hardware goal is to provide a light capable of being blinked by the software. A pure structural decomposition is used to divide the design into a mechanical portion and an electrical/electronic portion in order to accommodate the fact that two engineers from those two disciplines will be designing the system. A structure+goal decomposition is used on the mechanical portion to support the bulb (with a lamp holder socket) and protect the bulb (with a snap-on plastic cover).

On the software side, the goal is to blink the light when appropriate. In this case, a pure structural decomposition results in an object-oriented software architecture. A further pure behavioral decomposition of each object leads to the various object methods.

CONCLUSIONS

This paper proposes a taxonomy to organize different decomposition strategies and suggests strengths and weaknesses observed when applying the various approaches. Comprehensive methodologies should address the full breadth of design situations possible, including both technical and non-technical factors. In order to accomplish this, methodologies should accommodate multiple decomposition strategies in order to exploit the identified strengths while working around weaknesses. While some of the issues addressed may not seem appropriate in an ideal design methodology, they nonetheless exist in the real world, and must be accounted for when providing practitioners with useful tools.

Ad-hoc decomposition may help attain optimality, but may come at the considerable expense of circumventing methodical design practices and benefits. Future work will explore ways of incorporating the descriptive technique presented here into a prescriptive design methodology that permits approximating a globally optimal design, reaps the benefits of a methodical design process, and accommodates both business and technical issues.

REFERENCES

Albus, J. S., 1982, "RCS: a reference model architecture for intelligent control," *IEEE Computer*, Vol. 15, No. 5, pp. 56-59.

Amdahl, G. M., 1967, "Validity of the single processor approach to achieving large-scale computing capabilities," *1967 Spring Joint Computing Conf., AFIPS Conf. Proc.*, Vol. 30, pg. 483.

Blanchard, B. S., and Fabrycky, W. J., 1989, *Systems Engineering and Analysis*, Prentice Hall.

Britton, K. H., and Parnas, D. L., 1981, *A-7E Software Module Guide*, NRL Memorandum Report 4702, Naval Research Laboratory, Washington DC.

Chi, M. T. H., Slotta, J. D., and de Leeuw, N., 1994, "From things to processes: a theory of conceptual change for learning science concepts," *Learning and Instructions*, Vol. 4, pp. 27-43, Elsevier Science Ltd.

Hemenway, M. W., 1989, "Functional Analysis of Weapon Systems," *Logistics Spectrum*, Spring 1989, pp. 19-23.

Marks, M. D., Eubanks, C. F., and Ishii, K., 1993, "Life-cycle clumping of product designs for ownership and retirement," *Design Theory and Methodology - DTM `93*, pp. 83-89.

MIL-STD-499B, Draft, 1992, *Draft Military Standard for Systems Engineering*, U.S. Department of Defense.

Pahl, G., and Beitz, W., 1984, *Engineering Design*, Wallace, K. (ed.), The Design Council, London.

Pugh, S., 1990 *Total Design: integrated methods for successful product engineering*, Addison-Wesley.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W., 1991, *Object-Oriented Modeling and Design*, Prentice Hall.

Suh, N. P., 1990, *The Principles of Design*, Oxford University Press.