

Modular System Synthesis: Example for Composite Packaged Software

Mark Sh. Levin, *Senior Member, IEEE*

Abstract—The paper describes a framework for synthesis of a composite system (packaged software) on the basis of interconnected system components when the components and their interconnection (compatibility) are evaluated upon ordinal coordinated scales. Hierarchical multicriteria morphological design is used including generation of design alternatives (DAs) for system components and their combining into the target system while taking into account weighted interconnectability among DAs. This composition problem is based on building a morphological clique. Composite decisions are analyzed by a quality vector which involves the ordinal effectiveness of components and their interconnection. A numerical example demonstrates the hierarchical design (integration) of the system, improvement of obtained design decisions, and multistage design.

Index Terms—Combinatorial optimization, modular design, multicriteria analysis, packaged software, selection of software, system analysis, system refinement, system synthesis.

I. INTRODUCTION

IN RECENT decades, the significance of complex systems design has increased. In our opinion, the following methods can be listed as basic ones.

- 1) Formal methods for design [15], [20], [43], [64].
- 2) Optimization on the basis of complex mixed integer nonlinear programming (e.g., design/synthesis in chemical engineering) [39], [47].
- 3) Non-linear multicriteria (multiobjective) optimization [84], [118], [119].
- 4) Multidisciplinary optimization (mainly in aerospace and structural engineering) [4].
- 5) Parameter space investigation (PSI) for the system design and financial planning [113], [118].
- 6) Various methods of global optimization [40], [41].
- 7) Hierarchical system design [14], [50], [62], [80], [127] and modular system design [11], [53], [73], [131] including hierarchical morphological multicriteria design (HMMD) [72], [73].
- 8) Design on the basis of grammar description for composable systems [42], [105], [109].
- 9) Special artificial intelligence approaches on the basis of expert systems (knowledge-based systems), e.g., R1/MICON [48], [83], [120] for computer engineering, VLSI design, etc. [15], [44].
- 10) Hybrid methods [120].

Manuscript received February 21, 2000; revised September 17, 2002 and April 14, 2004. The preliminary version of the paper was presented at IBM-Users School (Tel-Aviv, February 1991). This paper was recommended by Associate Editor J. Miller.

The author is at P.O. Box 102, Moscow 117208, Russia (e-mail: hslevin04@yahoo.com).

Digital Object Identifier 10.1109/TSMCC.2004.840065

At the same time, it is reasonable to point out two main design problems which are crucial for complex systems [4], [15], [20], [44], [64], [73].

- 1) Basic system design:
 - 1.1) Design of a system as a wholeness.
 - 1.2) Synthesis or integration of a composite system.
- 2) System reengineering, improvement.

In this paper, a modular approach HMMD [72], [73] is used for system synthesis (integration) as a combinatorial hierarchical selection and composition design of a complex software package consisting of multiple components. Our approach is illustrated on the basis of a numerical example: synthesis of composite packaged software. We consider the following problems: integration of the system, improvement of obtained design decisions, and multistage design.

Several perspectives of software development processes are presented in [9] and [10]. Computer-aided support environments (CASE) are often used for software development [115]. These integrated environments are defined as collections of special software and hardware tools including program composition techniques for automatic software generation. However, in this case, components are, in the main, operators or program parts.

Further, it is reasonable to examine three system levels of software engineering as follows.

- 1) Bottom-level coding (coding software blocks; coding complex software).
- 2) Intermediate-level modular design (modular design of composite software, modular design of user interfaces).
- 3) High-level design (joint design of a system consisting of software and hardware components; strategic planning of product life cycle).

In our opinion, the significance of levels 2 and 3 increases [5], [10], [11], [81], [90], [91], [96]. Contemporary software development is often based on combining many software systems and packages. For example, we face this situation when modular programming is used for software development [11], [81], [87], [90], [91]. Many modern software packages have standard interfaces for the connection with other software packages. Thus, a system developer can combine selected packaged software components into a target composition. It is reasonable to point out the possibility for the end user to integrate his/her software environment on the basis of standard components too. Moreover, this process is an integration of systems, including both software and hardware components [96]. Special new CASE tools are developed for coordinating the complex projects [125]. Issues of interconnectability of software components are crucial ones in the design of hybrid systems [52], [81], [120], concurrent en-

engineering environments [97] and design research projects [30], [86]. Note that the combinatorial approach is very important for the design of a composite applied software environment on the basis of well-known software packages.

In recent years, the multicriteria selection problem has been played a central role in design processes [19], [32], [56], [63], [69]. Various approaches have been used for software selection, for example: a) criteria like cost/benefit [1], [8], and [111]); b) empirical models [46]; c) mathematical models [110]; d) optimization knapsack-type models for reliability of modular software systems [6], [11]; and (e) multicriteria methodology [65] including analytic hierarchy process (AHP) [8], [33], [107]. Note also the selection and implementation of packaged software for small businesses are analyzed in [55]. The significance of software quality, including both product and process quality, is pointed out in [9].

In our approach, an objective function of the composition problem is represented as a quality vector consisting of two parts: 1) quality of system components and 2) quality of compatibility between the system components. The paper involves a brief description of software development, our HMMD, and several numerical examples.

II. APPROACHES TO SOFTWARE DEVELOPMENT

Various well-known software development approaches have been used including: structured and system approaches, the Jackson approach to software development (JSD), object-oriented programming, component-based software development, and soft systems methodology [10], [21], [29], [90], [104], [114], [126], [127]. Many papers are oriented to the systematic comparison of software design methods [22], [54], [116]. Typical phases in the complete software engineering life cycle are described in detail by Beam *et al.* [10]. Software development process control is investigated in [5]. Some papers included specific paradigm for software development, e.g., stages of software storming [57]; steps of an iterative design for DSS [117]; a knowledge-based approach to automate a software design method for concurrent systems is described in [85]; and phases of expert system development [51]. Note a survey on automating software development is contained in [85]. Roberts *et al.* have examined factors of implementing issues for system development methodology on the basis of an analysis of 61 companies [100]. Ivri and Maansaari have published a review of concepts for system development methods [54]. Aspects of software adaptability are described in [34]. A social analysis of software development paradigms on the basis of three approaches (traditional, iterative, and component-based) is described in [101]. Planning and management of software evolution processes are examined in [24], [25], and [67]. For example, Lehman and Ramil have analyzed eight laws of software evolution [67].

The main concepts of software development are based on the following two approaches: 1) hierarchical design [3], [16], [21], [126], [127], [129]; and 2) prototyping [17]. Here the first approach is applied. In this paper, we assume the use of modular programming when module versions are developed and assessed independently, and there are some independently evaluated interfaces between module versions [design alternatives

(DAs)] for different software functions. Note version models for software configuration management are examined many years [28] including several types of basic models. Our approach corresponds to the *composition model*.

III. HIERARCHICAL MORPHOLOGICAL DESIGN

The extended description of HMMD for composite systems is described in [72] and [73]). HMMD is similar to some discipline-independent approaches based on decision making, creation and knowledge-based technology, e.g., morphological analysis [131], AHP [103], structured design [127], object oriented development [14], and design methodology of Carnegie Mellon University [30], etc.

A. Assumptions

The following basic assumptions are taken into account:

- 1) The designed system can be decomposed into a tree-like structure, i.e., a designed system is hierarchical and modular.
- 2) At each hierarchical level, the examined effectiveness of the system (subsystem) can be represented as an aggregation of two independent parts: effectiveness of components, and effectiveness of compatibility among the components.
- 3) Monotone criteria (Cr) of the following types are used for the system components: a) additive, b) multiplicative, and c) supreme.
- 4) The effectiveness of the subsystems interconnection (Is) is an aggregation of all independent pairwise interconnection between subsystem pairs. Here we use ordinal scale of pairwise compatibility (0, ... 5; 5 corresponds to the best quality, 0 corresponds to impossible interconnection).
- 5) Multicriteria descriptions (without compatibility) of DAs can be transformed into ordinal effectiveness (priority) $r \in [1, \dots, k]$, (1 corresponds to the best quality; here $k = 4$), and all these priority scales are coordinated.

B. Scheme

HMMD consists of information elements (the tree-like model of the designed system and design module), procedures, and user(s). The design module corresponds to a node of the hierarchical model and includes the following: 1) Cr; 2) DAs and their estimates on Cr; 3) estimates of Is between DAs of different components (morphological classes); 4) constraints for composite DAs; and 5) resultant information (priorities of DAs, etc.).

A generalized scheme of HMMD is as follows:

- Phase 1) Top-Down design of the system model (i.e., design of tree-like system model, specification of Cr, specification of constraints for composite DAs if necessary).
- Phase 2) Generation of DAs for leaf nodes of the model.
- Phase 3) (iterative) Bottom-Up hierarchical selection and composition: evaluation of DAs based upon Cr; multicriteria comparison of DAs and definition of their priorities; specification of Is among DAs;

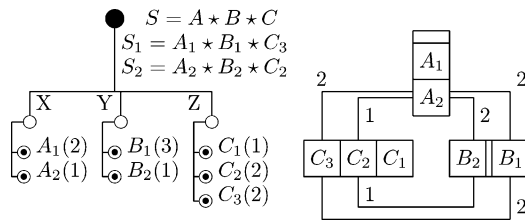


Fig. 1. Example (priorities of DAs are shown in brackets).

and composition of DAs for the next higher hierarchical level.

- Phase 4) Analysis and refinement of composite DAs (revelation of bottlenecks, forming improvement actions).

HMMD involves the following three basic problems: 1) multicriteria selection/ranking of DAs; 2) combining the DAs (the composition or synthesis problem); and 3) analysis/refinement of the composite DAs.

C. Multicriteria Selection

Various techniques for multicriteria selection are available [18], [31], including: 1) multiattribute utility analysis [38], [59]; 2) multicriterion decision making [61]; 3) AHP [103]; 4) outranking techniques [102]; 5) special knowledge bases [78]; 6) neural networks [124]; and 7) hybrid techniques [35]. In our example, DSS COMBI was applied for multicriteria ranking [71], [73]. Results using other techniques are expected to be similar.

D. Composition Problem

Now consider the composition/synthesis problem.

Find a composition (morphological scheme, morphological clique) $S = S(1) * \dots * S(i) * \dots * S(m)$ (where $S(i)$ is a selected design alternative for the i th system component) of DAs (one representative for each morphological class) with nonzero Is.

Similar problems (e.g., multiple choice problem, nonlinear Boolean programming, problem of compatible representatives, morphological clique) have been examined by a number of authors [7], [11], [60], [82], [112], [131]. Here the following quality vector for resultant composite decision S is used: $N(S) = (w(S); n(S))$ where $w(S)$ is the minimal value of pairwise compatibility in S , and $n(S) = (n(1), \dots, n(r), \dots, n(k))$ where $n(r)$ is the number of selected DAs of the r th quality in S . Thus, we search for solutions which are nondominated by $N(S)$. In addition, we may take into account some constraints as monotone functions on parameters of S , which play a role of filtering. (Our examples do not include constraints.) The composition scheme consists of the following two stages: 1) construction of feasible morphological schemes, as in [60], [112], and [131]; and 2) selection of Pareto-effective solutions based on N . This combinatorial problem is NP-hard in general [60]. It is reasonable to use an enumerative algorithm starting from the maximal value of $w = 5$ and from the best DAs at each morphological class. Dynamic programming method can also be applied [75].

Fig. 1 illustrates decomposable system $S = A * B * C$. Here examples of the composite system are: $S_1 = A_1 * B_1 * C_3$ and

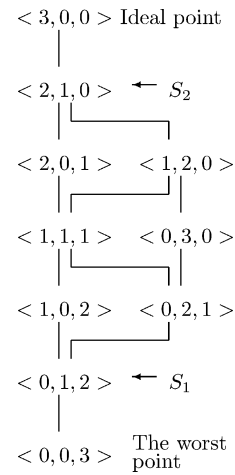


Fig. 2. Position (histogram) presentation of the lattice of system quality for $N = (w; n(1), n(2), n(3))$, $w = \text{const}$, $m = 3$, $l = 3$.

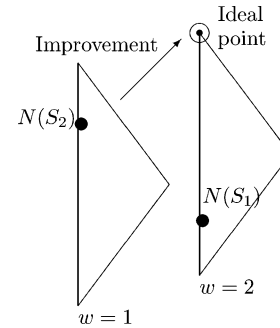


Fig. 3. Discrete space of system quality for $N(S)$.

$S_2 = A_2 * B_2 * C_2$. For composite decision in Fig. 1, we get $N(S_1) = (2; 0, 1, 2)$ and $N(S_2) = (1; 2, 1, 0)$.

Thus, $N(S_1)$ and $N(S_2)$ are Pareto-effective points. Fig. 2 depicts an example of the discrete space of system quality for a fixed level of compatibility [for $n(S)$]. Fig. 3 depicts the integrated discrete space of system quality (for $N(S)$) and examples of decisions. Note the space consists of two ordered lattices each of them corresponds to the lattice from Fig. 2.

E. Analysis and Improvement of Composite Decisions

We examine the following standard types of solution elements (DAs, Is) with respect to a composite decision S : S -improving, S -neutral, and S -aggravating ones by vector $N(S)$. These elements are similar to *strengthening-weakening* graph nodes investigated by Harary *et al.* [49]. In our approach, S -aggravating solution elements are considered as *bottlenecks*.

Note we apply the same viewpoint both for DAs and for Is. Mainly, it is assumed that we examine an *improvement action* when S -aggravating solution element (DAs or Is) is enhanced only on 1. Let a composition S be a quasi-solution, if the only one *improvement action* by $n(S)$ or $w(S)$ transforms S into an element of the Pareto-effective point set (by N). Searching for the quasi-solutions can be implemented by finding a corresponding neighborhood of the Pareto-effective point set. In the same way, we can introduce the neighborhood of the *ideal* solution(s) (consisting of the best elements by DAs and Is). It can be

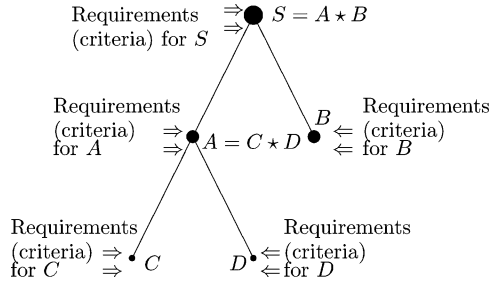


Fig. 4. Hierarchy of requirements.

shown that an algorithm for finding the elements of the neighborhood for any set $\{S\}$ is polynomial in $|\{S\}|$. Thus, it is reasonable to examine for the refinement S -aggravating elements for Pareto-effective DAs, and above-mentioned quasi-solutions.

A reasonable approach is to examine a series of solution sets as follows (an ordinal scale for system excellence): 1) the *ideal* solution; 2) the Pareto-effective points by N ; 3) the neighborhood of the Pareto-effective points; and 4) the worst feasible solution.

In addition, we can consider constraints for composite DAs. Thus, we can execute some *improvement actions* of the following types: a) constructing the *ideal* solution(s); b) refining the Pareto-effective points on the basis of examining the S -aggravating elements; and c) refining the quasi-solutions to extend the Pareto-effective point set.

IV. EVALUATION OF SOFTWARE

Many authors have investigated criteria for the evaluation of software [5], [12], [13], [18], [33], [37], [79], [94], [98], [106], [121], [123], [127]–[129]. The evaluating concepts for software quality include the following [13], [99]: 1) quality objectives; 2) quality factors; 3) criteria; 4) evaluation process; 5) measures; and 6) aggregation measures.

Specific levels are proposed by Regina and De Rocha [98]: objectives, factors, and subfactors. Lee uses AHP [103] as a structured methodology for software evaluation [66]. In this approach, some organizational aspects are taken into account and the following levels are investigated: manager, personnel, programs, and modules. Card *et al.* consider software quality engineering which involves quality measures for all design stages as follows: requirements, design, coding, testing, and maintenance [23]. Sprague and Carlson analyze specifically DSSs and their key characteristics [117]. The number of known software metrics exceeds 200 [36], [130]. Phan examines mainly software quality as minimization of the number of faults or defects that exist during and even after software development [94]. Metrics for in-process tracking and measurement have been examined in [58].

In our case, it is necessary to design a special hierarchical criteria space. Note a similar approach is the basic one in requirements engineering [3]. Fig. 4 illustrates a hierarchy of requirements (criteria) for a system and its parts.

V. ASSESSMENT OF INTERCONNECTION

A measure of interconnection (compatibility) among modules in a program structure (coupling) depends on the inter-

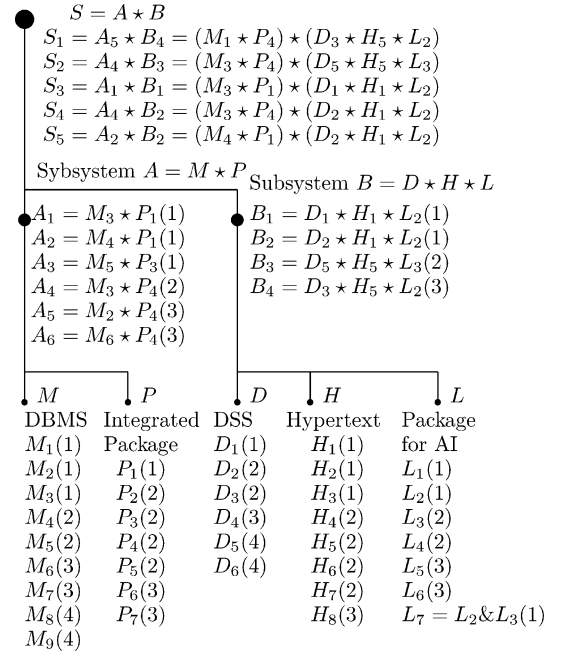


Fig. 5. Structure of system (priorities of DAs are shown in brackets).

faces between modules [95]. Software systems contain several types of interrelation among components, including: data, control, and sequencing [108]. Evaluation processes of the interconnection can be based on the following factors [81], [96], [108], and [129]: 1) interface between software components; 2) data format; and 3) common elements (i.e., program modules, element of interfaces). Liebowitz considered the following three levels of the hierarchy for hybrid systems: a) loose coupling (communication via data files and parameters passing); b) tight coupling (direct communication); and c) fully integrated [81]. Ziegler introduced an ordinal scale $(0, \dots, 9)$ for modules coupling [129, p. 114]. Our specification of Is weights in HMMD is based on the following two approaches: a) expert judgment; and b) multicriteria ranking the basic estimates of Is parameters above.

VI. EXAMPLE

A. Model of Composite Applied Software

Our example has a hypothetical character. We propose an integration of a composite IBM PC software environment for the end user consisting of several well-known packages. The following tree-like structure of composite software is analyzed (Fig. 5).

1) Subsystem $A = M * P$:

Database management system DBMS (M) + Integrated software IP (P).

2) Subsystem $B = D * H * L$:

Decision support system DSS (D) + hypertext system HS (H) + language for artificial intelligence LAI (L).

It is reasonable to point out the following sources to describe examined specific software packages: a) DBMS s (M) [88], [92], [93], [123]; b) IP's (P) [89], [123], [128]; c) DSSs (D) [18], [45], [71], [123]; d) HSs (H) [2], [27], [71]; and (e) LAI (L) [26], [68].

TABLE I
AGGREGATIVE CRITERIA

Criteria	Specification
F_{a1} Cost (-1)	$F_{m5} + F_{p7}$
F_{a2} Graphics	$\min(F_{m3}, F_{p1})$
F_{a3} Easy of use	$\max(F_{m1}, F_{p2})$
F_{a4} Telecommunication	$\min(F_{m2}, F_{p4})$
F_{a5} Adaptability of interface	$\min(F_{m4}, F_{p6})$
F_{a6} Text processor	$\max(F_{m3}, F_{p5})$
F_{a7} Report development	$\max(F_{m3}, F_{p2})$
F_{a8} Speed	$\max(F_{m2}, F_{p3})$
F_{a9} Environment for query building	F_{m6}
F_{b1} Cost (-1)	$F_{d7} + F_{h5} + F_{l1}$
F_{b2} Range of use	$\min(F_{h7}, F_{l3})$
F_{b3} AI components	$\max(F_{d5}, F_{h4})$
F_{b4} Experience of use	$F_{d6} + F_{h6} + F_{l2}$
F_{b5} Dialogue support	F_{d2}

B. Hierarchy of Criteria

Criteria and estimates of software are taken mainly from several journals ("PC World," "Microcomputers," "InfoWorld," "Expert Systems Users," etc.). We use the following criteria.

- 1) DBMSs: 1) tools of development (0 corresponds to "No," 1 corresponds to "Yes," and 2 corresponds to "Excellent"; further ordinal scale are similar); 2) speed (0, 1, 2); 3) report development (0, 1, 2); 4) adaptability of interface (0, 1); 5) cost (USD); and 6) query building environment (0, 1, 2).
- 2) IPs: 1) graphics (0, 1, 2); 2) easy of use report (0, 1, 2); 3) database (0, 1); 4) telecommunication (0, 1, 2); 5) text processor (0, 1, 2); 6) adaptability of interface (0, 1); and 7) cost (USD).
- 3) DSSs: 1) user interface (1, 2); 2) dialog support (0, 1, 2); 3) training subsystem (0, 1); 4) models (0, 1, 2); 5) AI components (0, 1); 6) previous experience of use (0, 1); and 7) cost (USD).
- 4) HSs: 1) browsing (0, 1); 2) graphics (0, 1); 3) multiple windows mode (0, 1); 4) AI components (0, 1); 5) cost (USD); 6) previous experience (0, 1); and 7) range of use (0, 1, 2).
- 5) LAIs: 1) cost (USD); 2) previous experience of use (0, 1, 2); 3) range of users (0, 1); and 4) quality of service (0, 1, 2).

Table I contains aggregative criteria for the subsystems (the first index corresponds to the component, negative monotony is shown in brackets).

C. Comparison of Initial Alternatives for Components

At this stage, the following operations for terminal vertices of the system model are executed: 1) generation of DAs for each leaf vertex; 2) assessment of DAs upon criteria; and 3) ranking the DAs upon criteria estimates. DAs for terminal vertices, and their estimates are shown in Tables II–VI. Note that computed resultant ordinal priorities of DAs are presented in Fig. 5.

TABLE II
ESTIMATES OF DAs

DA's	Criteria					
	1	2	3	4	5	6
M_1	2	2	2	2	482	3
M_2	2	2	2	2	490	3
M_3	1	1	1	1	486	2
M_4	1	1	1	1	237	2
M_5	1	1	1	1	502	3
M_6	1	1	1	1	451	0
M_7	1	1	1	1	333	0
M_8	0	0	0	0	240	0
M_9	0	0	0	0	180	0

TABLE III
ESTIMATES OF DAs

DA's	Criteria						
	1	2	3	4	5	6	7
P_1	2	2	1	3	1	1	506
P_2	2	2	1	0	2	0	94
P_3	2	2	3	1	2	0	438
P_4	1	1	1	1	1	1	477
P_5	2	0	1	1	1	1	510
P_6	1	2	0	1	0	0	130
P_7	0	2	1	0	0	0	87

TABLE IV
ESTIMATES OF DAs

DA's	Criteria			
	1	2	3	4
L_1	1700	1	2	2
L_2	3000	1	2	2
L_3	2500	2	1	2
L_4	5000	0	1	2
L_5	2850	1	1	1
L_6	13570	0	1	2
$L_7 = L_2 \& L_3$	5500	2	2	2

TABLE V
ESTIMATES OF DAs

DA's	Criteria						
	1	2	3	4	5	6	7
D_1	1	1	1	2	0	1	100
D_2	1	1	1	1	0	1	495
D_3	2	1	0	1	1	0	299
D_4	1	1	0	2	0	0	500
D_5	1	0	0	1	0	0	29
D_6	1	0	1	1	0	0	400

D. Composing of Subsystem Alternatives

Now let us examine the next hierarchical level and execute the following actions: 1) specification of compatibility between DAs; 2) composition off composite DAs; 3) assessment of DAs upon criteria; and 4) ranking the DAs. Compatibility estimates of DAs are presented in Tables VII and VIII. DAs of subsystems A and B are shown in Table IX. Fig. 6 depicts a concentric presentation of composite DAs B_1 and B_3 .

TABLE VI
ESTIMATES OF DAs

DA's	Criteria						
	1	2	3	4	5	6	7
H_1	1	1	1	1	350	0	2
H_2	1	1	1	1	600	1	2
H_3	1	1	1	1	300	1	1
H_4	1	1	1	0	500	1	0
H_5	1	0	1	1	700	0	1
H_6	0	0	1	0	100	0	1
H_7	0	0	0	0	100	1	1
H_8	1	1	1	1	470	1	1

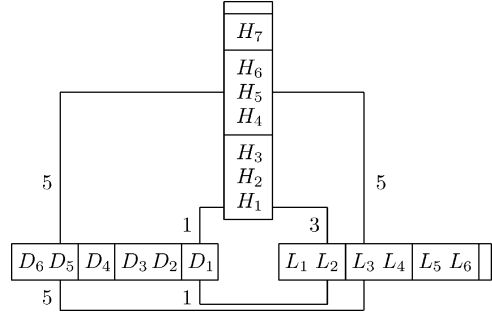


Fig. 6. Concentric presentation of composite DAs.

TABLE VII
COMPATIBILITY

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
M_1	0	0	2	3	0	0	0
M_2	0	0	1	1	0	0	0
M_3	2	2	1	3	0	0	0
M_4	3	0	0	0	0	0	0
M_5	0	0	4	0	0	0	0
M_6	0	0	0	5	0	0	0
M_7	0	0	0	0	0	0	0
M_8	0	0	0	0	0	0	0
M_9	0	2	1	0	0	0	0

TABLE X
COMPATIBILITY

	A_1	A_2	A_3	A_4	A_5	A_6
B_1	2	1	0	0	1	0
B_2	1	2	0	3	0	0
B_3	0	0	0	4	0	0
B_4	0	0	0	0	5	0

TABLE VIII
COMPATIBILITY

	D_1	D_2	D_3	D_4	D_5	D_6	L_1	L_2	L_3	L_4	L_5	L_6	L_7
H_1	1	3	0	0	0	0	0	3	0	0	0	0	0
H_2	0	0	0	0	0	0	0	0	0	0	0	0	0
H_3	0	0	0	0	0	0	1	1	1	2	2	2	1
H_4	0	0	0	0	0	0	0	0	0	1	1	1	0
H_5	1	0	4	0	5	0	0	4	5	1	1	1	4
H_6	0	0	0	0	0	0	0	0	0	1	1	1	0
H_7	0	0	0	0	0	0	0	0	0	1	1	1	0
H_8	0	0	0	0	0	1	0	0	0	0	0	0	0
D_1							1	1	1	2	2	2	1
D_2							0	3	0	0	0	0	0
D_3							0	4	0	0	0	0	0
D_4							0	0	0	0	0	0	0
D_5							0	0	5	0	0	0	0
D_6							0	0	0	1	1	1	0

TABLE XI
DAs OF SYSTEM

Composite DA's	N
$S_1 = A_5 \star B_4 = (M_1 \star P_4) \star (D_3 \star H_5 \star L_2)$	5; 0, 0, 2, 0
$S_2 = A_4 \star B_3 = (M_3 \star P_4) \star (D_5 \star H_5 \star L_3)$	4; 0, 2, 0, 0
$S_3 = A_1 \star B_1 = (M_3 \star P_1) \star (D_1 \star H_1 \star L_2)$	2; 2, 0, 0, 0
$S_4 = A_4 \star B_2 = (M_3 \star P_4) \star (D_2 \star H_1 \star L_2)$	3; 1, 1, 0, 0
$S_5 = A_2 \star B_2 = (M_4 \star P_1) \star (D_2 \star H_1 \star L_2)$	2; 2, 0, 0, 0

target system whit respect to the criteria, and to compare. Further, one may analyze some bottlenecks and improvement actions (Table XII). In our example, all improvement actions correspond to the improvement of Pareto-effective points. Note that the improvement of interconnection (A_1, B_1) or (A_2, B_2) from 2 to 5 allows to get the following two *ideal* decisions: $A_1 \star B_1$ and $A_2 \star B_2$. Results of this analysis can be applied as initial information for redesign. Fig. 7 depicts the discrete space of system excellence and Pareto-effective decisions.

TABLE IX
DA'S OF SUBSYSTEMS

DA's	Criteria								N	
	1	2	3	4	5	6	7	8		9
$A_1 = M_3 \star P_1$	992	2	2	3	1	1	4	3	2	2; 2, 0, 0, 0
$A_2 = M_4 \star P_1$	743	2	2	3	0	1	2	3	2	3; 1, 1, 0, 0
$A_3 = M_5 \star P_3$	940	2	2	1	0	2	3	3	3	4; 0, 2, 0, 0
$A_4 = M_3 \star P_4$	963	1	1	1	1	1	4	3	2	3; 1, 1, 0, 0
$A_5 = M_1 \star P_4$	959	1	1	1	1	1	4	2	3	3; 1, 1, 0, 0
$A_6 = M_6 \star P_4$	928	2	1	1	0	1	1	3	0	5; 0, 1, 1, 0
$B_1 = D_1 \star H_1 \star L_2$	3450	2	1	2	1					1; 3, 0, 0, 0
$B_2 = D_2 \star H_1 \star L_2$	3845	2	1	2	1					3; 2, 1, 0, 0
$B_3 = D_5 \star H_5 \star L_3$	3229	1	1	2	0					5; 0, 2, 0, 1
$B_4 = D_3 \star H_5 \star L_2$	3999	1	1	1	1					4; 1, 2, 0, 0

E. Composition and Analysis of System Alternatives

Compatibility estimates between DAs of the subsystems are shown in Table X. Table XI contains composite DAs for the target system. This allows to assess the composite DAs of the

VII. REDUNDANCY OF ALTERNATIVES

Usually, redundancy of system components is examined for increasing the system reliability [11], [122]. In our case (software), a system with redundancy may actually provide an improvement with respect to of usability because it implies the availability of alternative software packages. Let us examine a numerical example for subsystem B with the use of the following alternative (two-element redundancy) $L_7 = L_2 \& L_3$. We consider the aggregative estimates with respect to the 1) sum, 2) maximum, and 3) minimum (Table I) of element estimates.

The compatibility estimate of L_7 equals the minimum of corresponding estimates of the involved DAs (Table VIII). The priority of L_7 will be equal to 1. Finally, we get the composite decision as follows: $D_1 \star H_5 \star L_7$ with $N = (1; 1, 2, 0, 0)$. This decision is not an element of the Pareto-effective set, but it can be examined as a point for the improvement (Table XII). Fig. 8 depicts our composite decision with redundancy.

TABLE XII
BOTTLENECKS AND IMPROVEMENT ACTIONS

Composite DA's	Bottlenecks		Actions w/r
	DA's	Is	
$A_1 = M_3 \star P_1$	M_4	(M_3, P_1)	$2 \Rightarrow 3$
$A_2 = M_4 \star P_1$			$2 \Rightarrow 1$
$A_2 = M_4 \star P_1$		(M_4, P_1)	$3 \Rightarrow 4$
$A_3 = M_5 \star P_3$		(M_3, P_4)	$2 \Rightarrow 3$
$A_3 = M_5 \star P_3$		M_5	$2 \Rightarrow 1$
$A_3 = M_5 \star P_3$	P_3		$2 \Rightarrow 1$
$A_4 = M_3 \star P_4$		P_4	$2 \Rightarrow 1$
$A_4 = M_3 \star P_4$	P_4	(M_3, P_4)	$3 \Rightarrow 4$
$A_5 = M_1 \star P_4$			$2 \Rightarrow 1$
$A_5 = M_1 \star P_4$		(M_1, P_4)	$3 \Rightarrow 4$
$A_6 = M_6 \star P_4$	P_4		$2 \Rightarrow 1$
$A_6 = M_6 \star P_4$		M_6	$3 \Rightarrow 2$
$B_2 = D_2 \star H_1 \star L_1$	D_2		$2 \Rightarrow 1$
$B_3 = D_5 \star H_5 \star L_3$	H_5		$2 \Rightarrow 1$
$B_3 = D_5 \star H_5 \star L_3$	L_3		$2 \Rightarrow 1$
$B_4 = D_3 \star H_5 \star L_2$	D_3		$2 \Rightarrow 1$
$B_4 = D_3 \star H_5 \star L_2$	H_5		$2 \Rightarrow 1$
$S_1 = A_5 \star B_4$	A_5		$3 \Rightarrow 2$
$S_1 = A_5 \star B_4$	B_4		$3 \Rightarrow 3$
$S_2 = A_4 \star B_3$	A_4		$2 \Rightarrow 1$
$S_2 = A_4 \star B_3$		B_3	$2 \Rightarrow 1$
$S_2 = A_4 \star B_3$	A_4	(A_4, B_1)	$4 \Rightarrow 5$
$S_3 = A_1 \star B_1$		(A_1, B_1)	$2 \Rightarrow 3$
$S_4 = A_4 \star B_2$			$2 \Rightarrow 1$
$S_4 = A_4 \star B_2$	A_4	(A_4, B_2)	$3 \Rightarrow 4$
$S_5 = A_2 \star B_2$			$2 \Rightarrow 3$
$D_1 \star H_5 \star L_7$	H_5		$2 \Rightarrow 1$

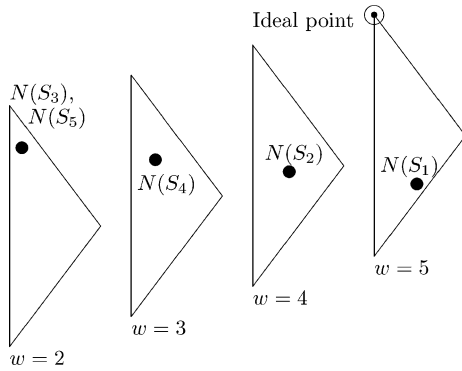


Fig. 7. Space of system quality and Pareto-effective points.

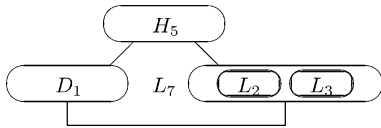


Fig. 8. Composite decision with redundancy.

VIII. MULTISTAGE DESIGN

Several versions of a multistage modular design on the basis of HMMD are described in [73] and [76]. Note this approach differs from traditional software evolution processes in [24] and [67]. Combinatorial evolution of composite systems on the basis of combinatorial system changes and HMMD is proposed in [77]. In this section, a three-stage modular design for subsystem B is examined. It is assumed that the compatibility estimates between DAs are improved (step-by-step) at each stage. The following two methods are applied:

- 1) **Aggregation Method:** An aggregation of the compatibility estimates and searching for a composite decision on the basis of the aggregated (average) estimates.
- 2) **Trajectory Method:** Two-level hierarchical solving process consisting of two phases as follows:
 - 2.1) Solving the design problem (synthesis of composite subsystem B) at each stage.
 - 2.2) Composing the composite decisions of the previous phase (for each stage) into a series composite decision (trajectory).

We consider the following improvement of compatibility:

- (H_1, D_1) : 2 (stage 2), the aggregated compatibility estimate equals 2;
- (H_1, D_2) : 4 (stage 2), 5 (stage 3), the aggregated compatibility estimate equals 2;
- (H_2, D_1) : 3 (stage 2), 5 (stage 3), the aggregated compatibility estimate equals 3;
- (H_5, D_3) : 5 (stage 2), the aggregated compatibility estimate equals 5;
- (H_5, D_5) : 5 (stage 2), the aggregated compatibility estimate equals 5;
- (H_1, L_2) : 4 (stage 2), 5 (stage 3), the aggregated compatibility estimate equals 4;
- (H_2, L_2) : 2 (stage 2), the aggregated compatibility estimate equals 2;
- (H_5, L_2) : 5 (stage 2), the aggregated compatibility estimate equals 5;
- (D_1, L_2) : 2 (stage 2), 3 (stage 3), the aggregated compatibility estimate equals 2;
- (D_2, L_2) : 4 (stage 2), 5 (stage 3) the aggregated compatibility estimate equals 4;
- (D_2, L_3) : 3 (stage 2), 4 (stage 3), the aggregated compatibility estimate equals 3;
- (D_3, L_2) : 5 (stage 2), the aggregated compatibility estimate equals 5; and
- (D_5, L_3) : 5 (stage 2), the aggregated compatibility estimate equals 5.

A. Aggregation Method

Here we solve the composition problem for the aggregated compatibility estimates. The resultant composite DAs are the following:

- a) $B_1^a = D_3 \star H_5 \star L_2$, $N(B_1^a) = (5; 1, 2, 0, 0)$;
- b) $B_2^a = D_1 \star H_1 \star L_2$, $N(B_2^a) = (2; 3, 0, 0, 0)$;
- c) $B_3^a = D_1 \star H_2 \star L_2$, $N(B_3^a) = (2; 3, 0, 0, 0)$.

Thus, the improvement with respect to interconnection leads to a new Pareto-effective composite decision B_3^a .

B. Trajectory Method

For the first stage we can use Pareto-effective composite DAs for subsystem B from Table IX as follows:

- $$B_1^1 = D_1 \star H_1 \star L_2, N(B_1^1) = (1; 3, 0, 0, 0);$$
- $$B_2^1 = D_2 \star H_1 \star L_2, N(B_2^1) = (3; 2, 1, 0, 0);$$
- $$B_3^1 = D_5 \star H_5 \star L_3, N(B_3^1) = (5; 0, 2, 0, 1);$$
- $$B_4^1 = D_3 \star H_5 \star L_2, N(B_4^1) = (4; 1, 2, 0, 0).$$

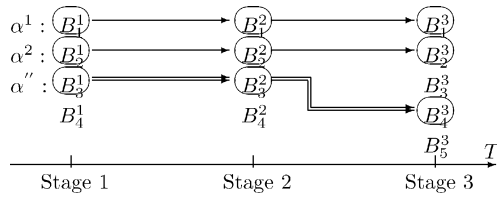


Fig. 9. Illustration for trajectory method.

The composite DAs for stage 2 are the following:

a) Pareto-effective decisions:

$$B_1^2 = D_1 \star H_1 \star L_2, N(B_1^2) = (2; 3, 0, 0, 0);$$

$$B_2^2 = D_2 \star H_1 \star L_2, N(B_2^2) = (4; 2, 1, 0, 0);$$

$$B_3^2 = D_3 \star H_5 \star L_2, N(B_3^2) = (5; 1, 2, 0, 0);$$

b) near Pareto-effective decisions (they are dominated by the above-mentioned Pareto-effective DAs):

$$B_4^2 = D_5 \star H_5 \star L_3, N(B_4^2) = (5; 0, 2, 0, 1).$$

The composite DAs for stage 3 are the following:

a) Pareto-effective decisions:

$$B_1^3 = D_1 \star H_1 \star L_2, N(B_1^3) = (2; 3, 0, 0, 0);$$

$$B_2^3 = D_2 \star H_1 \star L_2, N(B_2^3) = (5; 2, 1, 0, 0);$$

$$B_3^3 = D_1 \star H_2 \star L_2, N(B_3^3) = (2; 3, 0, 0, 0);$$

b) near Pareto-effective decisions (they are dominated by the above-mentioned Pareto-effective DAs):

$$B_4^3 = D_3 \star H_5 \star L_2, N(B_4^3) = (5; 1, 2, 0, 0);$$

$$B_5^3 = D_5 \star H_5 \star L_3, N(B_5^3) = (5; 0, 2, 0, 1).$$

Note “near optimal” composite DAs may be useful from the following viewpoint. A change of components into a trajectory can require some efforts, and it necessary to solve an additional top-level composition problem as follows: *to combine a trajectory while taking into account quality of composite DAs at each stage and a cost of the component changes (e.g., retraining of users, change of file formats).*

Here we obtain the following basic trajectories (without the changes): $\alpha_1 = \langle B_1^1, B_1^2, B_1^3 \rangle$, $\alpha_2 = \langle B_2^1, B_2^2, B_2^3 \rangle$. The example of the trajectory with the component change is the following: $\alpha' = \langle B_1^1, B_1^2, B_3^3 \rangle$ with the change $H_1 \rightarrow H_2$. Another strategy can be based on an improvement of a component. For example, the improvement of H_5 at stage 3 will lead to the following trajectory: $\alpha'' = \langle B_3^1, B_3^2, B_4^3 \rangle$. Fig. 9 illustrates the trajectory method.

IX. CONCLUSION

We have examined the modular system synthesis on the basis of three system design problems: 1) integration of the system, 2) improvement of obtained design decisions, and 3) multistage design. We have used extended version of HMMD (selection/composition of system components and improvement of system elements). In our opinion, our combinatorial approach (i.e., combinatorial engineering) can be very useful for the system design and redesign. An additional important goal is oriented to education processes. Let us list some significant possible future investigations:

- 1) an analysis of composite systems with the use of poset-type scales and fuzzy estimates of DAs including forecasting of future needs;
- 2) a study of dependency of DAs estimates and compatibility estimates;

- 3) an usage of the proposed approach for multidisciplinary systems (e.g., software engineering, algorithm design, databases, marketing); and
- 4) an analysis of software system evolution processes on the basis of our combinatorial approach.

ACKNOWLEDGMENT

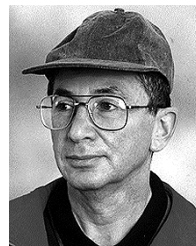
The author would like to thank I. Amihud (Herzelyia, Israel) and M. Bar-Ilan (Tel-Aviv, Israel) for their aid in 1991. The author would also like to thank the anonymous referees for their careful consideration of the paper and useful comments.

REFERENCES

- [1] N. Ahituv and M. Igbaria, “A model to facilitate cost, pricing and budget of computer services,” *Inform. Manag.*, vol. 14, pp. 235–241, 1988.
- [2] R. M. Akscyn, D. L. McCracken, and E. A. Yoder, “KMS: A distributed hypermedia system for managing knowledge in organizations,” *Commun. ACM*, vol. 31, no. 7, pp. 820–825, 1988.
- [3] M. Alford, “SREM at the age of eight: The distributed computing design system,” *Computer*, vol. 18, no. 4, pp. 36–46, 1985.
- [4] N. M. Alexandrov and M. Y. Hussaini, Eds., *Multidisciplinary Design Optimization: State of the Art*. Philadelphia, PA: SIAM, 1997, vol. 80.
- [5] J. D. Arthur, R. E. Nance, and O. Balci, “Establishing software development process control: Technical objectives, operational requirements, and the foundational framework,” *J. Syst. Softw.*, vol. 22, no. 2, pp. 117–128, 1993.
- [6] N. Ashrafi and O. Berman, “Optimization models for selection of programs, considering cost and reliability,” *IEEE Trans. Reliability*, vol. 41, no. 2, pp. 281–287, Feb. 1992.
- [7] R. U. Ayres, *Technological Forecasting and Long-Time Planning*. New York: McGraw-Hill, 1969.
- [8] J. F. Bard, A. de Silva, and A. Bergevin, “Evaluating simulation software for postal service use: Technique versus perception,” *IEEE Trans. Eng. Manag.*, vol. 44, no. 1, pp. 31–32, Jan. 1997.
- [9] V. R. Basili and J. D. Musa, “The future engineering of software: A management perspective,” *IEEE Computer*, vol. 24, no. 9, pp. 90–96, 1991.
- [10] W. R. Beam, J. D. Palmer, and P. A. Sage, “Systems engineering for software productivity,” *IEEE Syst., Man, Cybern.*, vol. 17, no. 2, pp. 90–96, Mar. 1987.
- [11] O. Berman and N. Ashrafi, “Optimization models for reliability of modular software systems,” *IEEE Trans. Softw. Eng.*, vol. 19, no. 11, pp. 1119–1123, Nov. 1993.
- [12] H. Bidgoli, “DSS products evaluation: An integrated framework,” *J. Data Manag.*, pp. 27–34, Nov. 1989.
- [13] B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. J. MacLeod, and M. J. Merrit, *Characteristics of Software Quality*. Redondo Beach, CA: TWR Systems and Energy, Inc., 1978.
- [14] G. Booch, *Object Oriented Development*. CA: Benjamin Cummings, 1991.
- [15] D. Braha and O. Maimon, *A Mathematical Theory of Design: Foundations, Algorithms and Applications*. Dordrecht, The Netherlands: Kluwer, 1998.
- [16] F. P. Brooks Jr., *The Mythical Man-Month. Essays on Software Engineering*. Reading, MA: Addison-Wesley, 1975.
- [17] R. Budde, K. Kantz, K. Kuhlenkamp, and H. Zullighoven, *Prototyping. An Approach to Evolutionary System Development*. Berlin, Germany: Springer-Verlag, 1991.
- [18] D. M. Buede, “Software review. Overview of MCDA software market,” *J. Multicriteria Dec. Anal.*, vol. 1, no. 1, pp. 59–61, 1992.
- [19] D. M. Buede and R. W. Choisser, “Providing an analytical structure for key system choices,” *J. Multicriteria Dec. Anal.*, vol. 1, no. 1, pp. 17–27, 1992.
- [20] D. M. Buede, *The Engineering Design of Systems: Models and Methods*. New York: Wiley, 1999.
- [21] J. Cameron, *JSP & JSD: The Jackson Approach to Software Development*, 2nd ed. Washington D.C.: IEEE Computer Society Press, 1989.
- [22] D. N. Card, “Software quality engineering,” *Inform. Software Technol.*, vol. 32, no. 1, pp. 3–10, 1990.
- [23] D. N. Card, F. E. McCarry, and G. T. Page, “Evaluating software engineering technologies,” *IEEE Trans. Softw. Eng.*, vol. 13, no. 7, pp. 845–851, Jul. 1987.

- [24] N. Chapin, J. E. Hale, K. M. Khan, J. F. Ramil, and W. G. Tan, "Types of software evolution and software maintenance," *J. Softw. Maint. Evol.: Res. Practice*, vol. 13, no. 1, pp. 1–30, 2001.
- [25] B. W. Chatters *et al.*, "Modeling a long term software evolution process," *J. Softw. Process – Improvement Practice*, vol. 5, no. 2/3, pp. 95–102, 2000.
- [26] F. Cohen and C. Guitfoyle, "Explore smalltalk and OPS," *Exp. Syst. User*, vol. 2, no. 11, pp. 10–11, 1987.
- [27] J. Conklin, "Hypertext: An introduction and survey," *IEEE Computer*, vol. 20, no. 9, pp. 17–41, 1987.
- [28] R. Conradi and B. Westfechtel, "Version models for software configuration management," *ACM Comput. Surv.*, vol. 30, no. 2, pp. 232–282, 1998.
- [29] T. DeMarco, *Structured Analysis and System Specification*. Englewood Cliffs, NJ: Prentice-Hall, 1979.
- [30] G. H. Demes, S. J. Fenves, I. E. Grossman, C. T. Hendrickson, T. M. Mitchell, F. B. Prinz, D. P. Siewiorek, E. Subrajmanian, S. Talukdar, and A. W. Westerberg, "The engineering design research center of carnegie mellon university," *Proc. IEEE*, vol. 81, no. 1, pp. 10–23, 1993.
- [31] J. S. Dyer, P. C. Fischburn, R. E. Steuer, J. Wallenius, and S. Zionts, "Multiple criteria decision making, multiattribute utility theory: The next ten years," *Manag. Sci.*, vol. 38, no. 5, pp. 645–654, 1992.
- [32] H. A. Eschenauer, "Structural optimization – A need in design process?," in *Engineering Optimization in Design Process*, H. A. Eschenauer, C. Mattheck, and N. Olhoff, Eds. Berlin, Germany: Springer-Verlag, 1991, vol. 63, Lecture Notes in Engineering, pp. 1–13.
- [33] K.-J. Farn and D.-H. Lo, "A method for selection of enterprise information system based on the combination of the analytic hierarchy process and the information system management planning," in *Proc. 10th Int. Conf. MCDM*, vol. III, Taiwan, R.O.C., 1992, pp. 221–236.
- [34] M. Fayad and M. P. Cline, "Aspects of software adaptability," *Commun. ACM*, vol. 39, no. 10, pp. 58–59, 1996.
- [35] K. Fedra, C. Zhao, and L. Winkelbauer, "Interactive multicriteria decision support: Combining rule-based and numerical approaches," in *User-Oriented Methodology and Techniques of Decision Analysis and Support*, J. Wessels and A. P. Wierzbicki, Eds. Berlin, Germany: Springer-Verlag, 1993, vol. 397, Lecture Notes in Economics and Mathematical Systems, pp. 48–64.
- [36] N. Fenton and S. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*. Boston, MA: PWS, 1997.
- [37] G. R. Finnie, G. E. Wittig, and D. I. Petkov, "Prioritizing software development productivity factors using the analytic hierarchy process," *J. Syst. Softw.*, vol. 22, no. 2, pp. 129–139, 1993.
- [38] P. C. Fishburn, *Utility Theory for Decision Making*. New York: Wiley, 1970.
- [39] C. A. Floudas, *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications (Topics in Chemical Engineering)*. New York: Oxford Univ. Press, 1995.
- [40] —, *Deterministic Global Optimization*. Norwell, MA: Kluwer, 1999.
- [41] C. A. Floudas and P. Pardalos, Eds., *State of the Art in Global Optimization: Computational Methods and Applications*. Norwell, MA: Kluwer, 1996.
- [42] D. Garlan and D. E. Perry, "Introduction to the special issue on software architecture," *IEEE Trans. Softw. Eng.*, vol. 21, no. 4, pp. 269–274, 1995.
- [43] J. Gero and F. Sudweeks, Eds., *Advances in Formal Design Methods*. Norwell, MA: Kluwer, 1996.
- [44] J. Gero and F. Sudweeks, Eds., *Artificial Intelligence in Design'98*. Dordrecht, The Netherlands: Kluwer, 1998.
- [45] B. L. Golden, A. Hevner, and D. Power, "Decision insight system for microcomputers: A critical evaluation," *Comp. Oper. Res.*, vol. 13, no. 2–3, pp. 287–300, 1986.
- [46] J. Griese and R. Kurpicz, "Investigating the buying process for the introduction of data processing in small and medium-sized firms," *Inform. Manag.*, vol. 8, pp. 41–51, 1985.
- [47] I. E. Grossmann, "Mixed-integer nonlinear programming techniques for the synthesis of engineering systems," *Res. Eng. Des.*, vol. 1, no. 2/3, pp. 205–228, 1990.
- [48] A. P. Gupta, W. P. Birmingham, and D. P. Siewiorek, "Automating the design of computer systems," *IEEE Trans. Computer-Aided Design of Integr. Circuits Syst.*, vol. 12, no. 4, pp. 473–487, Apr. 1993.
- [49] F. Harary, R. Z. Norman, and D. Cartwright, *Structural Models: An Introduction to the Theory of Directed Graphs*. New York: Wiley, 1965.
- [50] G. Harhalakis, C. P. Lin, R. Nagi, and J. M. Proth, "Hierarchical decision making in computer integrated manufacturing systems," in *Proc. 3rd Int. Conf. CIM*, 1992, pp. 15–24.
- [51] F. Hayes-Roth, D. A. Waterman, and D. Lenat, Eds., *Building Expert Systems*. Reading, MA: Addison-Wesley, 1983.
- [52] S. Hedberg, "New knowledge tools," *Bite*, vol. 7, pp. 104–111, 1993.
- [53] C. C. Huang and A. Kusiak, "Modularity in design of products and systems," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 28, no. 1, pp. 66–77, Jan. 1998.
- [54] J. Ivari and J. Maansaari, "The usage of system development methods: Are we stuck to old practice?," *Inform. Softw. Technol.*, vol. 40, no. 9, pp. 501–510, 1998.
- [55] M. A. Janson and A. Subramanian, "Packaged software: Selection and implementation policies," *INFOR*, vol. 34, no. 2, pp. 133–154, 1996.
- [56] J. C. Jones, *Design Methods*. New York: Wiley, 1981.
- [57] P. W. Jordan, K. S. Keller, R. W. Tucker, and D. Vogel, "Software storming. Combining rapid prototyping and knowledge engineering," *Computer*, vol. 22, no. 5, pp. 39–48, 1989.
- [58] S. H. Kan, J. Parrish, and D. Manlove, "In-process metrics for software testing," *IBM Syst. J.*, vol. 40, no. 1, pp. 220–241, 2001.
- [59] R. L. Keeny and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. New York: Wiley, 1976.
- [60] D. E. Knuth and A. Raghunathan, "The problem of compatible representatives," *SIAM J. Discr. Math.*, vol. 5, no. 3, pp. 422–427, 1992.
- [61] P. Korhonen, J. Wallenius, and S. Zionts, "Solving the discrete multiple criteria problems using convex cones," *Manag. Sci.*, vol. 30, no. 11, pp. 1336–1345, 1984.
- [62] P. S. Krasnoshekov, V. V. Morozov, and V. V. Fedorov, "Decomposition in design problems" (in Russian), *Eng. Cybern.*, no. 2, pp. 7–17, 1979.
- [63] N. Kuppuraju, P. Ittimakin, and F. Mistree, "Design through selection: A method that works," *Design Studies*, vol. 6, no. 2, pp. 91–106, 1985.
- [64] A. Kusiak, *Engineering Design: Products, Processes, and Systems*. New York: Academic, 1999.
- [65] A. Le Blanc and M. T. Jellassi, "DSS software selection: A multiple criteria decision methodology," *Inform. Manag.*, vol. 17, no. 2, pp. 49–66, 1989.
- [66] H. Lee, "A structured methodology for software development effort prediction using the analytic hierarchy process," *J. Syst. Softw.*, vol. 21, no. 2, pp. 179–186, 1993.
- [67] M. M. Lehman and J. F. Ramil, "Rules and tools for software evolution planning and management," *Ann. Softw. Eng.*, vol. 11, pp. 15–44, 2001.
- [68] P. S. Lehner and S. W. Barth, "Expert systems on microcomputers," *Expert Syst.*, vol. 2, no. 4, pp. 198–208, 1985.
- [69] R. R. Levary, "Introduction to engineering design using operations research methods," in *Engineering Design. Better Results Through Operations Research Methods*, R. R. Levary, Ed. New York: North Holland, 1988, pp. 1–16.
- [70] M. S. Levin, "Usage of discrete decision making problems for design of software," in *Communication of IBM-Users School*. Tel-Aviv, Israel: Kfar-Maccabia, 1991, pp. 47–51.
- [71] M. S. Levin, "Hierarchical components of human-computer systems," in *Human-Computer Interaction*, L. J. Bass, J. Gornostaev, and C. Unger, Eds. Berlin, Germany: Springer-Verlag, 1993, vol. 753, Lecture Notes in Computer Science, pp. 140–151.
- [72] —, "Hierarchical morphological multicriteria design of decomposable systems," *Concurrent Eng.: Res. Appl.*, vol. 4, no. 2, pp. 111–117, 1996.
- [73] —, *Combinatorial Engineering of Decomposable Systems*. Dordrecht, The Netherlands: Kluwer, 1998.
- [74] —, "Combinatorial selection and synthesis of composite packaged software," Faculty Business Admin., Univ. Ottawa, Ottawa, ON, Canada, Tech. Rep. no. 98-06, 1998.
- [75] —, "System synthesis with morphological clique problem: Fusion of subsystem evaluation decisions," *Inform. Fusion*, vol. 2, no. 3, pp. 225–237, 2001.
- [76] —, "Toward combinatorial analysis, adaptation, and planning of human-computer systems," *Appl. Intell.*, vol. 16, no. 3, pp. 235–247, 2002.
- [77] —, "Combinatorial evolution of composite systems," in *Proc. 16th Eur. Meeting Cybernetics System Research*, vol. 1, Austria, 2002, pp. 275–280.
- [78] H.-L. Li, "Solving discrete multicriteria decision problems based on logic-based decision support systems," *Decision Support Syst.*, vol. 3, pp. 101–118, 1987.
- [79] J. S. Liebman, "Implementation issues for operations research software," *Comp. Oper. Res.*, vol. 13, no. 2&3, pp. 347–358, 1986.
- [80] D. C. Liebisch and A. Jain, "JESSI COMMON FRAMEWORK design management – The means to configuration and execution of the design process," in *Proc. Eur. Design Automation Conf. EURO-DAC*, Los Alamitos, CA, 1992, pp. 552–557.

- [81] J. Liebowitz, "Roll your own hybrids," *Bite*, vol. 7, pp. 113–115, 1993.
- [82] S. Martello and P. Toth, *Knapsack Problem: Algorithms and Computer Implementation*. New York: Wiley, 1990.
- [83] J. McDermott, "R1: A rule-based configurator of computer systems," *Artif. Intell.*, vol. 19, no. 2, pp. 39–88, 1982.
- [84] K. Miettinen, *Nonlinear Multiobjective Optimization*. Norwell, MA: Kluwer, 1999.
- [85] K. L. Mills and H. Gomaa, "Knowledge-based automation of a design method for concurrent systems," *IEEE Trans. Softw. Eng.*, vol. 28, no. 3, pp. 228–255, Mar. 2002.
- [86] D. V. Morse and C. Hendrickson, "A communications model to aid knowledge-based design systems," *Artif. Intell. Eng. Design, Anal., Manuf. (AI EDAM)*, vol. 4, no. 2, pp. 99–115, 1990.
- [87] O. Nierstrasz, S. Gibbs, and D. Tschrititz, "Component-oriented software development," *Commun. ACM*, vol. 35, no. 9, pp. 160–165, 1992.
- [88] S. D. Palmer, "A new dimension for flat-file databases," *InfoWorld*, vol. 11, no. 5, pp. 45–51, 1989.
- [89] R. Pavlac, "All-in-one for the road," *InfoWorld*, vol. 12, no. 6, pp. 55–67, 1990.
- [90] F. Pena-Mora, S. Vadhavkar, and S. K. Dirisala, "Component-based software development for integrated construction management software applications," *Artif. Intell. Eng. Design, Anal., Manuf. (AI EDAM)*, vol. 15, no. 2, pp. 173–187, 2001.
- [91] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture," in *ACM SIGSOFT Softw. Eng. Notes*, vol. 17, 1992, pp. 40–52.
- [92] N. Petreley, Z. Banapour, and L. Slovic, "Analyzing relational databases," *InfoWorld*, vol. 12, no. 2, pp. 51–68, 1990.
- [93] N. Petreley, J. Duncan, L. Slovic, and Z. Banapour, "Relational power: Part II," *InfoWorld*, vol. 12, no. 26, pp. 56–61, 1990.
- [94] D. D. Phan, "Software quality and management," *Inform. Syst. Manag.*, vol. 18, no. 1, pp. 56–67, 2001.
- [95] R. Pressman, *Software Engineering*. New York: McGraw-Hill, 1987.
- [96] C. V. Ramamoorthy, C. Chandra, H. G. Kim, Y. C. Shim, and V. Vij et al., "Systems integration: Problems and approaches," in *Proc. 2nd Int. Conf. Systems Integration*, P.A. Ng et al., Eds., 1992, pp. 522–529.
- [97] Y. V. R. Reddy, K. Srinivas, V. Jagannathan, and R. Karinthi, "Computer support for concurrent engineering," *Computer*, vol. 26, no. 1, pp. 12–15, 1993.
- [98] A. Regina and C. Da Rocha, "Software quality assurance in HEP," *Comput. Phys. Commun.*, vol. 26, no. 1–3, pp. 524–527, 1989.
- [99] B. C. Reimann and A. D. Waren, "User-oriented criteria for the selection of DSS software," *Commun. ACM*, vol. 28, no. 2, pp. 166–179, 1985.
- [100] T. L. Roberts Jr., M. L. Gibson, K. T. Fields, and R. K. Rainer Jr., "Factors that impact implementing a system development methodology," *IEEE Trans. Softw. Eng.*, vol. 24, no. 8, pp. 640–649, Aug. 1998.
- [101] D. Robey, R. Welke, and D. Turk, "Traditional, iterative, and component-based development: A social analysis of software development paradigms," *Inform. Technol. Manag.*, vol. 2, no. 1, pp. 53–70, 2001.
- [102] B. Roy, "The outranking approach and foundations of ELECTRE methods," in *Readings in MultiCriteria Decision Aid*, C. A. Bana e Costa, Ed. Berlin, Germany: Springer-Verlag, 1990, pp. 155–183.
- [103] T. L. Saaty, *The Analytic Hierarchy Process*. New York: McGraw-Hill, 1980.
- [104] S. Sawyer and P. J. Guinan, "Software development: Processes and performance," *IBM Syst. J.*, vol. 37, no. 4, pp. 552–569, 1998.
- [105] C. Schaffer, "On the role of architecture in systems engineering," in *Computer Aided Systems Theory – EUROCAST'97*, F. Pichler and R. Moreno-Diaz, Eds. Berlin, Germany: Springer-Verlag, 1997, vol. 1333, Lecture Notes in Computer Science, pp. 13–33.
- [106] B. Schneiderman, *Software Psychology*. Cambridge, MA: Winthrop, 1980.
- [107] M. J. Schniederjans and R. L. Wilson, "Using the analytic hierarchy process and goal programming for information system project selection," *Inform. Manag.*, vol. 20, pp. 333–342, 1991.
- [108] R. W. Selby, "Interconnectivity analysis techniques or error localization in large systems," *J. Syst. Softw.*, vol. 20, no. 3, pp. 267–272, 1993.
- [109] M. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*. Englewood Cliffs, NJ: Prentice-Hall, 1996.
- [110] P. Shoval and U. Lugasi, "Models for computer systems evaluation and selection," *Inform. Manag.*, vol. 12, no. 3, pp. 117–129, 1987.
- [111] ———, "Computer systems selection: The graphical cost-benefit approach," *Inform. Manag.*, vol. 15, pp. 163–172, 1988.
- [112] J. Singhal and J. L. Katz, "A branch-and-fathom algorithm for the long range process design problem," *Manag. Sci.*, vol. 36, no. 4, pp. 513–516, 1990.
- [113] I. M. Sobol, "An efficient approach to multicriteria optimum design problems," *Surv. Mathem. Industry*, vol. 1, pp. 259–281, 1992.
- [114] I. Sommerville, *Software Engineering*. Reading, MA: Addison-Wesley, 1989.
- [115] ———, "Integrated project support environment (IPSE)," *Microprocess. Microsyst.*, vol. 13, no. 4, pp. 254–262, 1989.
- [116] X. Song and L. J. Osterweil, "Toward objective systematic design method comparisons," *IEEE Software*, vol. 9, no. 3, pp. 43–53, 1992.
- [117] R. H. Sprague Jr. and E. C. Carlson, *Building Effective Decision Support Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
- [118] R. B. Statnikov and J. Matusov, *Multicriteria Optimization and Engineering*. New York: Chapman and Hall, 1995.
- [119] R. E. Steuer, *Multiple Criteria Optimization: Theory, Computation, and Application*. New York: Wiley, 1986.
- [120] E. A. Sykes and C. C. White III, "Multiobjective intelligent computer-aided design," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 6, pp. 1498–1511, Nov.–Dec. 1989.
- [121] D. Van Tassel, *Program Style, Design, Efficiency, Debugging, and Testing*. Englewood Cliffs, NJ: Prentice-Hall, 1978.
- [122] F. A. Tillman, C.-L. Hwang, and W. Kuo, "Optimization techniques for system reliability with redundancy – A review," *IEEE Trans. Reliability*, vol. 26, no. 3, pp. 148–155, Sep. 1977.
- [123] E. Turban, *Decision Support System and Expert System: Management Support System*, 2nd ed. New York: Macmillan, 1990.
- [124] J. Wang and B. Malakooti, "A feedforward neural network for multiple criteria decision making," *Comp. Oper. Res.*, vol. 19, no. 2, pp. 151–167, 1992.
- [125] T. Williams, "New CAE tools aimed at coordinating complex projects," *Computer Design*, vol. 33, no. 5, pp. 65–68, 1994.
- [126] N. Wirth, *Systematic Programming. An Introduction*. Englewood Cliffs, NJ: Prentice-Hall, 1976.
- [127] E. Yourdon, *Techniques of Program Structure and Design*. Englewood Cliffs, NJ: Prentice-Hall, 1975.
- [128] A. L. Zeichik, "Integrated software," *InfoWorld*, vol. 11, no. 33, pp. 45–60, 1989.
- [129] C. A. Ziegler, *Programing System Methodologies*. Englewood Cliffs, NJ: Prentice-Hall, 1983.
- [130] H. Zuse, *Software Complexity: Measures and Methods*. New York: Walter de Gruyer, 1990.
- [131] F. Zwicky, *Discovery Invention, Research Through the Morphological Approach*. New York: McMillan, 1969.



Mark Sh. Levin (M'00–SM'05) received the M.S. degree in radioengineering from Moscow Technological University, Moscow, Russia, in communication and informatics, the M.S. degree in mathematics from Moscow State University, Moscow, and the Ph.D. degree in systems analysis from the Institute for System Analysis of Russia Academy of Sciences, Moscow, in 1970, 1975, and 1982, respectively.

For 20 years, he was with research and development organizations and the Academy of Sciences, Moscow. After that, he conducted research projects in Israel, Russia, Japan, and Canada. His research interests include systems engineering, decision making, combinatorial optimization, engineering education, and applications. He has authored over 60 research papers and several books.

Dr. Levin is a member of ACM, International Society on MCDM, International Society of Applied Intelligence, and ORSIS (Israel), and has served as a program committee member for several conferences.