V Jubileuszowa Wyższa Górska Międzynarodowa Szkoła PTI

Szczyrk, 21 - 25 czerwca 1993

Polskie Towarzystwo Informatyczne
Oddział Górnośląski

Katowice 1993

Druk: PWR Sp. z o.o. W-wa, ul. Żwirki i Wigury 81, tel. 22 24 26

V JUBILEUSZOWA WYŻSZA GÓRSKA MIĘDZYNARODOWA SZKOŁA PTI

WSPÓŁCZESNE OBSZARY ZASTOSOWAŃ INFORMATYKI

Szczyrk 21-25 czerwca 1993

Program

PONIEDZIAŁEK 1993.06.21

9^{45} **Otwarcie Szkoły**

10^{00} - 11^{00} **mgr inż. Wojciech Głazek** (DOKP Katowice): *Założenia, projekt i realizacja sieci transmisji danych KOLPAK.*

przerwa na kawę

11^{30} - 12^{15} **mgr inż. Włodzimierz Adamski** (WSK OBR PZL - MIELEC) *Sieci komputerowe w polskim przemyśle lotniczym.*

12^{30} - 13^{15} **mgr inż. Włodzimierz Adamski** (WSK OBR PZL - MIELEC): *Elektroniczna postać dokumentacji konstrukcyjnej i technologicznej.*

15^{00} - 16^{00} **mgr inż. Mariusz Klapper** (Załad Systemów Komputerowych i Oprogramowania "K&MSOFT"): *Praktyczne doświadczenia opracowywania i wdrażania w przedsiębiorstwach kompleksowych systemów informatycznych.*

przerwa na kawę

16^{20} - 17^{20} **dr inż. Jacek Stochlak** (ICL POLAND): *Czy naprawdę dinozaury - rola i miejsce dużych systemów komputerowych w systemach informatycznych przedsiębiorstw.*

17^{45} - 18^{45} **Zbigniew A. Nowacki** (AKREM, Łódź): *Szablony klas i listy elastyczne w pakiecie narzędziowym POLY do języka C.*

WTOREK 1993.06.22

9^{00} - 10^{30} **dr Jerzy Skrzypek** (Akademia Ekonomiczna): *Metodologiczne aspekty modelowania procesów gospodarczych (w ujęciu Dynamiki Systemowej).*

przerwa na kawę

11^{00} - 12^{30} **dr inż. Marek Miłosz** (Politechnika Lubelska) *SYMULACJA SYSTEMÓW - Symulacja procesów dyskretnych.*

15^{00} - 16^{00} **Artur Kasprzyk** (Uniwersytet Wrocławski): *Obiektowo-zorientowane analiza i projektowanie.*

przerwa na kawę

16^{20} - 17^{20} **mgr inż. Zbigniew Benedykt** (BEFAMA Bielsko-Biała): *Co daje obiektowe podejście do analizy i projektowania systemów informatycznych.*

17^{45} - 18^{45} **dr inż. Stanisław Kędzierski** (Akademia Ekonomiczna Katowice): *Modelowanie obiektów dynamicznych.*

ŚRODA 1993.06.23

9^{00} - 10^{00} **Marek Wierzbicki** (Politechnika Łódzka): *Tworzenie obiektów poprzez dziedziczenie mono- i polimorficzne.*

10^{15} - 11^{15} **mgr inż. Mirosław Wieczorek** (Politechnika Śląska) *Podstawowe zagadnienia programowania zorientowanego obiektowo - polimorfizm i dziedziczenie klas w języku C++.*

przerwa na kawę

11^{40} - 13^{00} **dr inż. Marek Średniawa** (Politechnika Warszawska): *Podejście obiektowe do specyfikacji na przykładzie języka CCITT SDL'92.*

15^{00} - 16^{00} **mgr inż. Roman Siminski** (Uniwersytet Śląski): *Projekt i realizacja obiektowo zorientowanego podsystemu komunikacji z użytkownikiem.*

przerwa na kawę

16^{20} - 17^{20} **Borys Stokalski, Andrzej Maciej Wierzba** (InfoVIDE): *LBMS Systems Engineer. Zintegrowane środowisko do tworzenia systemów informatycznych.*

17^{45} - 18^{45} **mgr inż Piotr Woźniak** (SuperMemo World): *SuperMemo as a new tool increasing the productivity of a programmer. A case study - Programming in Object Windows.*

CZWARTEK 1993.06.24

9^{00} - 10^{20} **dr Joanna Kurzok-Derda** (Instytut Systemów Sterowania Katowice): *System ekspertowy EKSSTER dla wspomagania projektowania i dokumentowania układów automatyki przemysłowej na bazie sterownika STERISS-2.*

przerwa na kawę

10^{40} - 12^{00} **mgr Danuta Kajrunajtys** (Akademia Ekonomiczna, Kraków): *Komputerowe systemy wspomagania decyzji a modele sytuacji decyzyjnych.*

12^{15} - 13^{00} **Prezentacja firmy Siemens-Nixdorf**

15^{00} - 16^{00} **prof. Wojciech M. Jaworski** (Concordia University, Montreal) *InfoMAPY - inna metoda gromadzenia wiedzy.*

przerwa na kawę

16^{20} - 17^{20} **dr Marian Kuraś** (Akademia Ekonomiczna, Kraków): *Komunikacyjne aspekty tworzenia systemów informacyjnych.*

17^{45} - 18^{45} **dr Andrzej Zaliwski** (Akademia Ekonomiczna, Kraków): *CASAD - nowe podejście do modernizacji SI.*

PIĄTEK 1993.06.25

10^{00} - 11^{15} **dr inż. Jan Baranowski** (THOMSON POLKOLOR): *Wykorzystanie baz danych opartych na Btrievie i NetwareSQL w środowisku WINDOWS.*

przerwa na kawę

11^{30} - 13^{00} **dr inż. Jacek Stochlak** (ICL POLAND): *Systemy przetwarzania transakcyjnego OLTP. Modele i standardy.*

20^{00} - skutku **Bal Prezesów**

POSIŁKI

Śniadanie	-	8^{30}
Obiad	-	13^{30}
Kolacja	-	19^{00}

Wojciech M. Jaworski
Computer Sc. Dept., Concordia University, Montreal

Andrzej Zaliwski, Marian Kuraś
Katedra Informatyki Akademii Ekonomicznej w Krakowie

INFOMAPY - inna metda gromadzenia wiedzy

Nie ma nic bardziej praktycznego
niż dobra teoria.
(Pan Edek powołujący się na W.I.L.
a ostatnio na A. Einsteina)

1. Wprowadzenie

Głównym problemem na jaki napotyka każdy, kto zajmuje się projektowaniem lub konserwacją wielkich systemów oprogramowania, jest ich złożoność. Systemy oprogramowania znacznie różnią się od innych systemów. Projektant musi operować na wielu różnych poziomach abstrakcji oraz zajmować się wieloma szczegółami. System oprogramowania - o czym należy pamiętać - przyjmuje złożoność swoich aplikacji. Oprócz tego sam proces wytwarzania oprogramowania jest szczególnie złożony. Typowymi produktami towarzyszącymi procesowi tworzenia oprogramowania są raporty, podręczniki, diagramy, teksty źródłowe itp. Do sporządzenia każdego z tych dokumentów są używane inne techniki notacyjne. Nierzadko do utworzenia pojedynczego dokumentu używa się kilku metod. W takiej sytuacji występuje silna zależność i pokrywanie się informacji w poszczególnych dokumentach, będących produktami kolejnych faz procesu rozwoju oprogramowania. Jest to nie do uniknięcia, zwłaszcza, że wspomniane dokumenty są rezultatem stosowania różnych podejść, metod, technik reprezentacji wizualnej, konwencji opisu, języków, narzędzi wspomagania itp. W związku z tym dokumenty związane z procesem rozwoju oprogramowania są fragmentarycznymi opisami i częściowo wzajemnie nakładającymi się na siebie obrazami systemu oprogramowania. Do dokumentowania SI stosowane są różne techniki diagramowania w celu prezentacji różnych aspektów funkcjonowania systemów (np. DFD - Data Flow Diagrams, ERD - Entity-Relationship Diagrams, STD - State-Transition Diagrams, Structure Charts, Jackson Data Structure Diagrams, Nassi-Shneiderman Diagrams). Każdy z diagramów może opisywać ten sam system i prezentuje jego komponenty. Żaden z nich nie reprezentuje całego systemu - daje tylko pewien specyficzny obraz, przekrój systemu. W konsekwencji modyfikacja jakiegoś komponentu musi spowodować odpowiednie zmiany w innych reprezentacjach. Takie pochodne zmiany nie zawsze są od razu jasne i czytelne. Utrzymanie spójności takiej dokumentacji wymaga znaczących nakładów pracy.

Tworzenie i rozwój oprogramowania możemy potraktować jako proces pozyskiwania wiedzy. Na określonym etapie tworzymy bazę wiedzy zawierającą informacje o tworzonym systemie. Rozwój następuje w wyniku pozyskiwania dalszych informacji i ich włączania do bazy wiedzy lub jako efekt przetwarzania informacji

zgromadzonej wcześniej. Przechowywana wiedza jest zróżnicowana w zależności od fazy projektu. Występuje na różnych poziomach szczegółowości począwszy od założeń programu a kończąc na kodzie. Rzadko zdarza się by powiązania między tymi poziomami były formalne lub kompletne. Przed techniką rozwoju oprogramowania stawia się następujące wymagania:

- formalnej notacji,
- technologii, która wspomaga i jest wspomagana przez notację,
- metod reprezentacji obiektów i wiedzy w obrębie danej dziedziny,
- reguł odwzorowania modeli pomiędzy różnymi konwencjami notacji.

Celem niniejszej publikacji jest przedstawienie techniki notacyjnej spełniającej powyższe wymagania, zatem mogącej stanowić podstawę rozwoju oprogramowania. Technika ta nosi nazwę: **'infoMapy'**. Jej możliwości pozwalają na opisywanie systemu oprogramowania na wielu poziomach abstrakcji oraz na uniknięcie trudności związanych z translacją i konwersją informacji projektowych, gdyż ta sama konwencja notacji może być użyta we wszystkich fazach procesu produkcji oprogramowania. Ponadto, notacja **infoMap** umożliwia modelowanie aplikacji i ich systemu oprogramowania na wszystkich poziomach rozwoju, począwszy od jego założeń i specyfikacji a kończąc na kodzie źródłowym programu.

2. infoMAPy jako technika notacji

W celu rozwinięcia formalizmu, który może być użyty dla szerokiego zakresu aplikacji na różnych poziomach abstrakcji, potrzebujemy podstawowych struktur i technik do manipulowania nimi. Notacyjnie technologia infoMap jest oparta na podstawowych pojęciach matematyki: zbiorach i relacjach. W sensie matematycznym zbiór jest kolekcją obiektów spełniających pewne warunki lub mających tę samą charakterystykę. Standardowymi strukturami matematyki dyskretnej są zbiory, funkcje, relacje i grafy. Relacje są szeroko używane w teorii baz danych, podczas gdy grafy są używane dla tworzenia narzędzi do rozwoju oprogramowania [15].

Relacja jest zdefiniowana jako odwzorowanie produktu kartezjańskiego pewnej liczby zbiorów na zbiór {0, 1}. Innymi słowy, relacja między zbiorami istnieje lub nie. Taka definicja relacji jest niewłaściwa dla wielu zastosowań deskryptywnych. W rzeczywistości bowiem relacja może także przenosić znaczenie. Przykładowo relacja "matka", matki do jej dziecka nie tylko istnieje ale także zawiera rodzaj zależności. Definicja relacji powinna więc zostać uogólniona jako odwzorowanie produktu kartezjańskiego pewnej liczby zbiorów na zbiór symboli, gdzie każdy symbol definiuje jednoznacznie rodzaj relacji. InfoMapa jest reprezentacją tak uogólnionej relacji.

Notacja infoMap [6]-[12] łączy wizualną prostotę grafu z gęstością informacyjną tablicy. Z dziedziny baz danych została zapożyczona koncepcja słownika danych. Reprezentacja wizualna została zapożyczona z dziedziny tablic decyzyjnych. Powstaje w ten sposób nowa, macierzopodobna reprezentacja o dużej pojemności informacyjnej.

Przeanalizujmy prosty przykład. InfoMapa z rysunku 1 przedstawia plan wykładu na temat infoMap. InfoMapa ta zawiera następujące informacje dotyczące:

- zastosowania przekroju,
- struktury wykładu,
- rysunków.

W jMapie wydzielone są dwie zasadnicze części:

- obszar definicji zbiorów
- obszar definicji relacji.

Kolumny 15 i 16 odpowiadają obszarowi definicji zbiorów, kolumny od 1 do 13 obszarowi definicji relacji.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1																	
2		A	A	A	A	A	A	A	A	A	A	A	A	A		{ZASTOSOWANIE PRZEKROJU}	
3		v														Instytucja	
4			v													Wykładowca	
5				v	v	v										Struktura wykładu	
6							v	v	v	v	v	v	v			Użycie rysunków	
7														v		Liczebność	
8		M	M	H	H	H	o	o	o	o	o	o	o	16		{CZĘŚCI WYKŁADU}	
9		v														**Cracow School of Economics**	
10		v														**Canadian -Polish Management Centre**	
11			v													Wojciech M. Jaworski, Concordia University, Montreal	
12				h			o									***infoMaps: a System for Knowledge Transfer Automation***	
13				1	h											Wprowadzenie	
14					1			o								Efektywność transferu wiedzy	
15					2				o							Dokladność transferu wiedzy	
16				2		h										Dlaczego "Automatyzacja transferu wiedzy"?	
17						1										Człowiek-, Komputer-, Wiedza "wykonawcza"	
18						2										Proces pozyskiwania wiedzy i kontrola jakości	
19						3										Zastosowanie transferu wiedzy	
20						4										Użycie wiedzy - Operator, Komputer	
21				3						o						Funkcjonalność "infoFarm"	
22				4							o					Produkcja oprogramowania - bardzo prosty przykład	
23				5								o				Składniki bazy wiedzy "infoFarm"	
24				6									o			Odzyskiwanie wiedzy	
25							M	M	M	M	M	M	M	8		{RYSUNKI}	
26							v									Model wykładu1	
27								v								Fig. A. Tryby transferu wiedzy	
28									v							Fig. B. "Captive Mind (wise joke)"	
29										v						Fig. 1. infoMap - przykład oceny studentów	
30										v						Fig. 2 .. 7.4 - "infoFarm"	
31											v					Program przeszukiwania binarnego: infoMapy 1 .. 12	
32												v				Przestrzeń Systemu: infoMapy 13 .. 17	
33													v			Przykład: MIS : infoMapa 18	
34																	
35		**Rys. 1. InfoMapa zawierająca plan wykładu n/t infoMAP**															
36																	

Nazwy zbiorów i ich elementy są zdefiniowane w obszarze definicji zbiorów. Relacje między zbiorami oraz relacje między elementami zbiorów są zdefiniowane w obszarze definicji relacji.

W obszarze definicji zbiorów, nazwy zbiorów są zestawione w tej samej kolumnie, ujęte w nawiasy klamrowe i napisane dużymi literami. Zdefiniowane w przykładzie z rysunku 1 nazwy zbiorów to: {ZASTOSOWANIE PRZEKROJU}, {CZĘŚCI WYKŁADU} oraz {RYSUNKI}. Elementy wchodzące w skład tych zbiorów są wymienione odpowiednio poniżej nazw tych zbiorów. Każda kolumna jMapy reprezentuje relację. Relacje są niepowtarzalne, z czego wynika, że nie powinno być dwóch identycznych kolumn. Dzięki temu unikamy redundancji danych. Każda komórka w obszarze definicji relacji odpowiadająca wierszowi elementu zbioru uczestniczy w danej relacji.

Uczestniczenie zbioru lub jego elementu w danej relacji jest oznaczane przez umieszczenie odpowiedniego elementu notacji (symbolu specjalnego) na skrzyżowaniu kolumny relacji z wierszem, w którym jest nazwa zbioru (lub nazwa elementu zbioru). Pusta komórka w jakimś miejscu kolumny relacji oznacza, że zbiór

lub jego element w odpowiadającym wierszu nie uczestniczy w tej relacji. Symbole specjalne są podzielone na dwie grupy *'Rola zbiorów'* oraz *'Rola elementów zbioru'*. Stosowane oznaczenia oraz zależności pomiędzy tymi grupami (infoSYNTAX) zostały zamieszczone na rysunku 10. Należy w tym miejscu zaznaczyć, że składnia infoMap jest otwarta. Użytkownik ma całkowitą swobodę w definiowaniu lub redefiniowaniu przyjętych oznaczeń, znaczenia obiektów infoMapy a także dopuszczalnych operacji na tych obiektach.

W jMapie istnieją dwa różne poziomy relacji. Pierwszy poziom specyfikuje relacje między zbiorami. Drugi poziom, niższy, relacje między elementami zbiorów. Pierwszy ze wspomnianych poziomów nosi nazwę **infoSCHEMy** *(infoFRAME)*. InfoSCHEMA jest modelową strukturą dla reprezentowania określonej dziedziny wiedzy. InfoSCHEMA stanowi ramową strukturę, która jest stopniowo wypełniana i uszczegóławiana aż do otrzymania infoMapy. Stąd infoMapy mogą być traktowane jako implementacja infoSCHEMy. Generalnie rzecz biorąc z jednej infoSCHEMy można otrzymać wiele infoMap. Podobnie jak w przypadku programów: na podstawie jednej specyfikacji można zakodować wiele programów. Przejście od infoSCHEMy do infoMapy wymaga udziału człowieka, podczas gdy operacja odwrotna może być przeprowadzana automatycznie.

Na rysunku 2 jest przedstawiona infoSCHEMA dla przykładu z poprzedniego rysunku. W przypadku interpretowania relacji w infoMapie należy najpierw analizować relacje między zbiorami (czyli relacje przedstawione na infoSCHEMie) a dopiero potem relacje między elementami zbiorów. W naszym przykładzie relacje zawartą w kolumnie 7 możemy interpretować w sposób następujący: *'Jeden (One) element zbioru {CZĘŚCI WYKŁADU} jest powiązany z wieloma (Many) elementami zbioru* {RYSUNKI}. Oznacza to, że między tymi zbiorami istnieje relacja typu 1:M. Ponadto czytamy z rysunku, że relacja w kolumnie 7 jest fragmentem relacji zagregowanej (Aggregation) wskazującej na autora tej kolumny/relacji wymienionego w zbiorze {AUTHOR}. Stopniowo, przez uzupełnianie infoSCHEMY o relacje między elementami zbiorów otrzymujemy infoMapę taką jak na rysunku 1.

Ważnym elementem każdej infoMapy jest specjalny zbiór o nazwie {VIEW} lub {VIEW's PURPOSE} (W wypadku rysunku 1 i 2 jest to {ZASTOSOWANIE PRZEKROJU}). Każda relacja, stanowiąca przecież określony przekrój dziedziny, jest opisana przez komentarze zawarte w tym zbiorze. Przykładowo na rysunku 1, w wierszu *'Struktura wykładu'* znajdują się trzy litery 'v' w kolumnach 4,5,6 oznacza to, że te kolumny opisują strukturę wykładu. Oprócz tego istnieje specjalny przekrój oznaczony jako *'cardinality'* (liczebność). Związana z nim kolumna zawiera informacje o liczebności zbiorów. Informacje te są przydatne przy weryfikacji mapy.

	1	2	3	4	7	14	15	16
1								
2		A	A	A	A	A		{ZASTOSOWANIE PRZEKROJU}
8		M	M	H	O	16		{CZĘŚCI WYKŁADU}
25					M	8		{RYSUNKI}
34								
35		**Rys. 2. InfoSCHEMA dla rysunku 1.**						
36								

Kolejny przykład - infoMapa przedstawiona na rysunku 3 zawiera zestawienie ocen dotyczących sytuacji w Polsce w 1992 roku, uzyskanych od różnych osób. Czytelnikom proponujemy samodzielne odczytanie informacji zawartych w tej infoMapie.

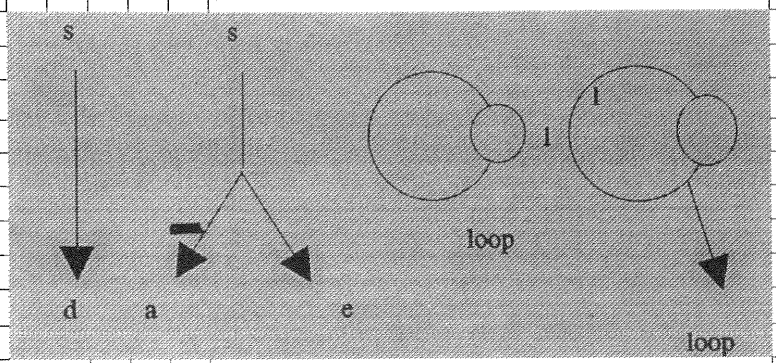
Jak już mieliśmy możliwość zauważyć, notacja infoMap jest notacją bardzo pojepojemną i elastyczną. W celu pokazania walorów tej techniki notacyjnej oraz przekonania, iż jest rzeczywiście notacją uniwersalną, mogącą opisywać każdy aspekt SI, w dalszej części niniejszej publikacji prezentujemy kilka przykładów. Większość z nich dotyczy samego programowania a jeden dziedziny od programowania bardzo odległej.

	1	2	3	4	5	6	7	8	9
1									
2									
3		A	A	A	A	A	A		{PUNKT WIDZENIA}
4		v	v		v				Polska widziana oczyma Polakow z Zachodu
5				v					Polska widziana oczyma Polakow w Polsce
6		v	v	v	v	v	v		Skala ocen; 1=wyjatkowo dobrze, 2=zupelnie dobrze, 3=troche dobrze, 4=neutralna, 5=troche zle, 6=zupelnie zle , 7=wyjatkowo zle
8		O	O	O	O	O	O		{OPINIODAWCA}
9		o							A. Targowski (na Zachodzie od 1980 r.)
10			o						J. Nowak Jezioranski (na Zachodzie od II Wojny Swiatowej)
11				o					S. Lem (w kraju)
12					o				W. Jaworski (na Zachodzie od 1968 r.)
13						o			Marian Kuras (roczne stypendium w Kanadzie 1991-92)
14							o		Andrzej Zaliwski (w kraju)
15		V	V	V	V	V	V		{DZIEDZINA}
16		6	3	6	6	5	5		Sytuacja polityczna
17		7	2	7	7	7	7		Strategia polityczna
18		6	2	6	6	4	5		Sytuacja gospodarcza
19		7	?	7	7	6	6		Strategia gospodarcza
20		?	1	?	5	4	4		Przywodcy/Dzialacze demokratyczni do 1989 r
21		6	5	6	6	6	5		Obecna kadra polityczna/politycy
22		6	?	5	6	6	7		Kultura polityczna
23		6	?	7	7	6	6		Wiedza polskiego spoleczenstwa o strategii gospodarczej
24		?	?	?	6	7	6		Wiedza polskiego spoleczenstwa o pozycji Polski w swiecie
25		?	?	?	6	3	4		Prognoza powaznych inwestycji/pozyczek zagranicznych
26									
27									
28		A	A	A	A	A	A		{ZRODLO}
29		v	v	v	v	v	v		K.M. Ford and all. ,"An Approach to Knowledge Acquisition Based on the Structure of Personal Construct Systems", IEEE Trans. on Knowledge and Data Eng., March 1991 pp 78-88
30		v	v	v	v	v	v		Z. Pawlak, "Rough Sets: Theoretical Aspects of Reasoning about Data", De Kuyper, Amsterdam, 1992
31		v							A. Targowski, "Rewolucja zjada wlasne dzieci", Studium Newsletter, January 1992
32			v						J. Nowak Jezioranski, "W odpowiedzi Andrzejowi Targowskiemu", Studium Newsletter, February/March 1992
33				v					S. Lem, "Swiat i Polska", Kultura, Paryz, Nr 5/536, 1992
34									
35									
104		**Rys. 3. Ocena sytuacji w Polsce w 1992 r. przedstawiona w postaci infoMAPY.**							
105					*Copyright © W.M. Jaworski 1992*				

3. Zastosowania w programowaniu

W celu ilustracji możliwości opisu przy zastosowaniu konwencji infoMap wybraliśmy sposób reprezentacji najbardziej elementarnych struktur sterujących i przykładów programów.

Właściwości elementarnych struktur sterowania są dyskutowane w wielu publikacjach (np. [2], [1], [3], [16]). Graf przepływu sterowania tych struktur sterujących może być reprezentowany przez cztery elementarne grafy oraz ich złożenie. Te podstawowe konstrukcje są określane jako: *'Sekwencja'*, *'Sekwencja z asercją* lub *wyjątkiem (exception)'*, *'Pętla'*, *'pętla z wyjściem'*. Rysunek 4a przedstawia te cztery elementarne konstrukcje a rysunek 4b sposób ich przedstawienia w postaci infoMap. Nazwy tych czterech podstawowych konstrukcji zostały wymienione jako elementy zbioru {GENERIC GRAPH}. Przecięcie wiersza, w którym jest wymieniona nazwa konstrukcji z kolumną opisującą tę konstrukcję jest oznaczone literą *'o'*. Litera *'s'* została użyta do określenia źródła (*source*) a litera *'d'* jako oznaczenie stanu docelowego (*destination*). Symbol *'l'* oznacza pętlę (*loop*), *'a'* - asercję (*assertion*), zaś *'e'* - wyjątek (*exception*).



branch					branch with fork

(a) Reprezentacja graficzna

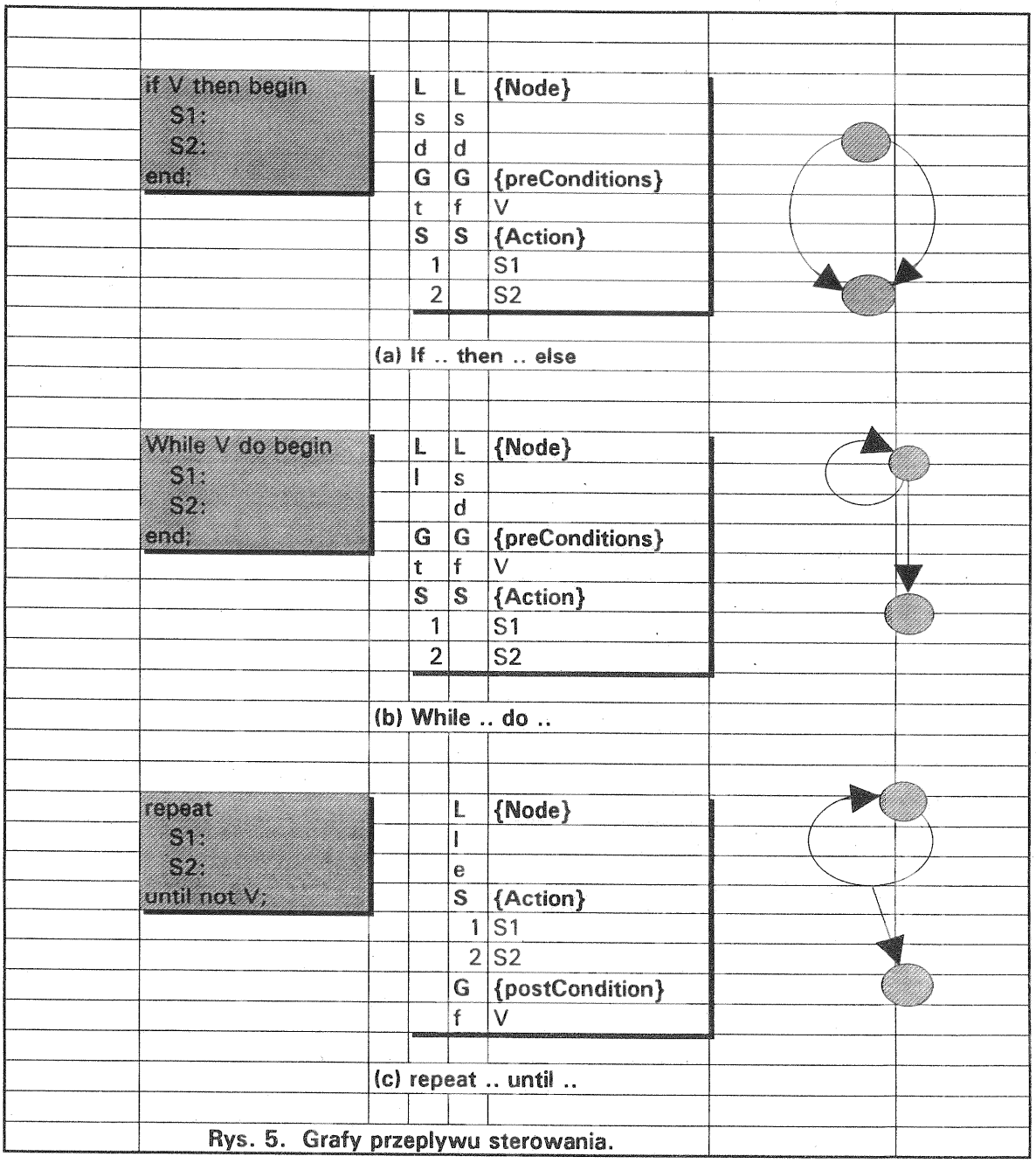
O	O	O	O		{GENERIC GRAPH}
o					branch from s to d
	o				branch from s to either a or e
		o			loop
			o		loop with exit to e
L	L	L	L		{NODE/STATE}
s					start
d					destination
	s				start
	a				asserted destination
	e				exception destination
		l			loop node
			l		loop node
			e		exception destination
A	A	A	A		{AUTHOR}
v	v	v	v		InfoSCHEMA Copyright © 1991 W.M. Jaworski
v	v	v	v		InfoMAP Copyright © 1991 W.M.Jaworski

(b) Reprezentacja w postaci infoMAPY

Rys. 4. Cztery podstawowe grafy przepływu sterowania.

Na rysunku 5 z lewej strony pokazano w postaci kodu strukturalnego zbliżonego do języka Pascal niektóre konstrukcje stosowane w programach. Środkowa część rysunku przedstawia te same konstrukcje w postaci infoMap. Zwróćmy uwagę na zbiór {PRECONDITION} na rysunku 5a. Jeśli wymieniony w nim warunek *'V'* jest spełniony (*true*), zachodzi sytuacja przedstawiona w pierwszej kolumnie. Jest wykonywana sekwencja akcji *S1* i *S2*. Litera *'S'* sygnalizuje operacje sekwencyjne. Cyfry *'1'* i *'2'* pod literą *'S'* określają wówczas kolejność wykonywania operacji. Jeśli warunek *'V'* nie jest spełniony (*false*), to mamy sytuację opisaną w drugiej kolumnie. Tak jak w poprzednim przypadku również następuje przejście z jednego stanu do drugiego ale akcje *S1* i *S2* nie mają miejsca.

Na rysunku 6 pokazano kod strukturalny dla programu przeszukiwania binarnego (a) oraz odpowiadającą mu reprezentację w postaci infoMapy (b). Pozycje *'O'*, *'L'*, *'G'*, *'S'* i *'G'* określają role jakie zostały przydzielone odpowiednim zbiorom tj. {*Transition*}, {*State*}, {*preCondition*}, {*Action*} oraz {*postCondition*}. Znaczenie poszczególnych pozycji jest następujące:



Rys. 5. Grafy przeplywu sterowania.

```
1: found:=false
2: while( first <= last)
    and  not  found do begin
              i := ( first + last ) div 2;
3:            if a[i]<X then first := i+1
4:            else if a[i]>X then last:=i-1
                    else found := true;
    end;
5: if a[i] = X then
            X found at i
    else X not found;
6:
```

(a) kod strukturalny

1	2	3	4	5	6	7	8	9	10	11	12	
O	O	O	O	O	O	O	O	O	9	{Transition}		
L	L	L	L	L	L	L	L	L	6	{State}		
s											1:	
d	s	s	d		d	d					2:	
	d		s	s							3:	
				d	s	s					4:	
							s	s			5:	
							d	d			6:	
G	G	G	G	G	G	G	G	G	4	{preCondition}		
	t	f									first<=last and not found	
			t	f							a[i] < X	
					t	f					a[i] > X	
							t	f			a[i] = X	
S	S	S	S	S	S	S	S	S	7	{Action}		
1											found := false	
	1										i:=(first + last) div 2	
			1								first:=i+1	
					1						last:=i-1	
						1					found := true	
							1				X found at i	
								1			X not found	

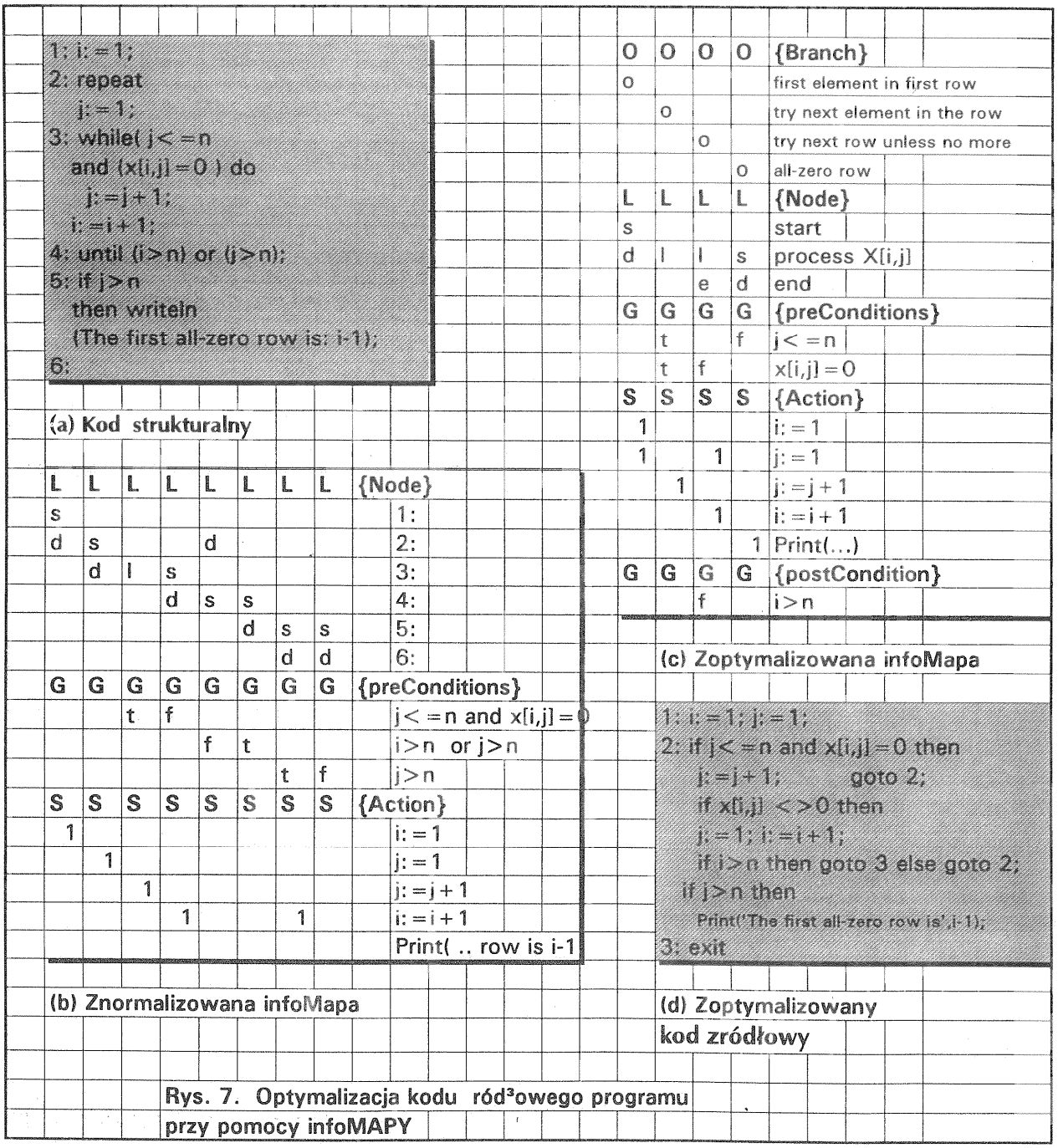
(b) Kod z rysunku (a) zapisany jako infoMAPA

Rys. 6. Algorytm wyszukiwania binarnego.

Każde (*each*) przejście (*transition*) łączy (*Links*) stany (*States*). Każde przejście jest zaimplementowane jako sekwencja (*Sequence*) akcji kontrolowanych (*Guarded*) przez warunki wstępne (*preconditions*) oraz kontrolowana (asserted) przez warunki końcowe (*postconditions*).

Litera *'A'* pozwala użytkownikowi na zaznaczenie faktu, że kilka kolumn jest zgrupowane w relację poziomą. Litera *'S'* oznacza relację pionową i sekwencyjne uporządkowanie elementów zbioru.

Poniżej zamieszczamy przykład oparty na następującym zadaniu, które sformułował Dijkstra w [4]: "Niech X będzie macierzą NxN zawierającą liczby całkowite. Należy napisać program który wyświetla numer



Rys. 7. Optymalizacja kodu ród³owego programu przy pomocy infoMAPY

pierwszego zerowego wiersza macierzy X, o ile taki istnieje."

Rozwiązanie tego problemu pochodzące z [15] i jego translację na infoMapę przedstawia rysunek 7 (a) i (b). Przez przetworzenie wierszy i kolumn infoMapy otrzymano zoptymalizowaną infoMapę (c) oraz zoptymalizowany kod źródłowy (d). Inne rozwiązanie tego samego problemu [13] oraz rozwiązania o nazwach 'Pancode' i 'EPN' zestawiono w postaci infoMapy na rys. 8. Rozwiązanie 'Pancode' reprezentowane jako infoMapa (nie zamieszczona) ma 4 węzły/stany, 6 rozgałęzień/przejść, 3 warunki wstępne i 5 akcji. Oryginalne rozwiązanie 'EPN' ma 5 węzłów/ stanów, 6 rozgałęzień/przejść, 3 warunki wstępne, 5 akcji i 1 warunek końcowy. Rozwiązanie w języku Pascal zaczerpnięte z [15] reprezentowane w postaci infoMapy (rysunek (b) ma 6 węzłów/ stanów, 8 rozgałęzień/przejść, 3 warunki wstępne i 5 akcji. Rozwiązanie zoptymalizowane (c) ma 3 węzły/stany, 4 rozgałęzienia/ przejścia, 2 warunki wstępne, 5 akcji

{Wersja}	A	A	A	A					
Zoptymalizowana	v								
Pancode		v							
EPN-long			v						
EPN-short				v					
{Atrybuty}	V	V	V	V					
branch	2	5	5	5					
branch with fork	0	0	1	0					
loop	1	1	0	1					
loop with exit	1	0	0	0					
state	3	4	5	4					
transition	4	6	6	6					
preCondition	2	3	3	3					
action	5	5	5	5					
postCondition	1	0	1	0					

Rys. 8. Zestawienie różnych rozwiązań zadania

i 1 warunek końcowy.

Rozwiązanie zostało zoptymalizowane - otrzymane przez połączenie stanów, przejść, przez eliminację redundantnych warunków i akcji.

Kompletna infoMapa z rysunku 9 zawiera zestawienie dwóch algorytmów przeszukiwania binarnego. Zostały one oznaczone *'Binary Search Program Version 1'* i *'Binary Search Program Version 2'* jako elementy zbioru {PROCESS}. Kolumny oznaczone literami *'v'* w wierszu *'Binary Search Program Version 1'* opisują ten algorytm. Analogiczna sytuacja zachodzi w przypadku drugiego algorytmu.

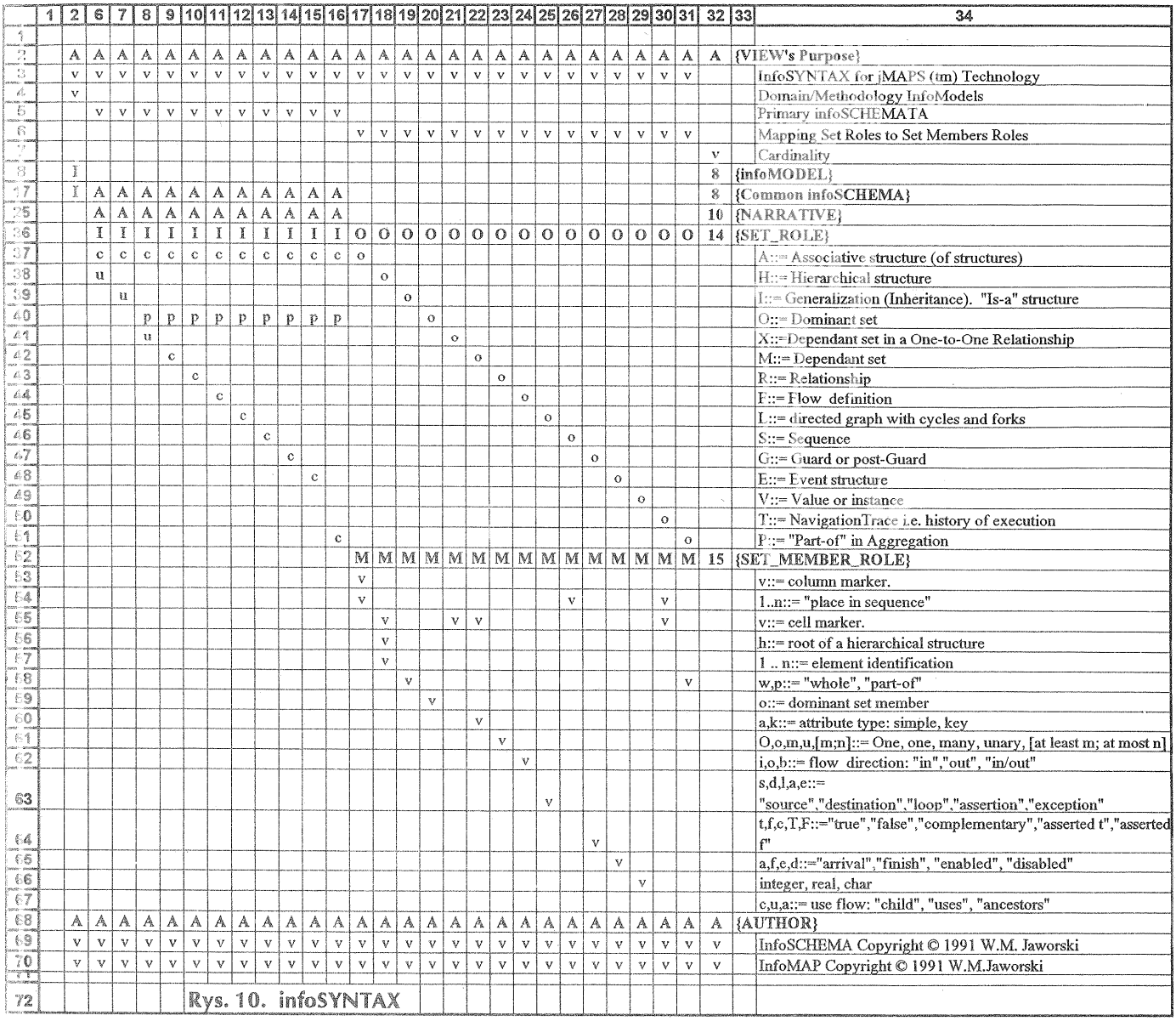
InfoMapę możemy zacząć analizować praktycznie w dowolnym miejscu w zależności od tego co nas interesuje. Zacznijmy może od zbioru {*preCONDITION*}. Zawarty w nim jest warunek *'first<=last'* i jeśli jest on spełniony (tzn. w jego wierszu jest litera *'t'*), analizujemy kolumnę 3, jeśli nie (*'f'*) - kolumnę 4. Zakładamy, że warunek jest spełniony *('t')* i analizujmy kolumnę 3. Idąc w górę kolumny napotykamy litery *'s'* i *'d'*, które mówią, że następuje przejście ze stanu *'1:Start'* do stanu *'2:Process element a[i]'* (Przetwarzanie elementu a[i]). Jeszcze wyżej w zbiorze {*TRANSITION*} badanej kolumnie odpowiada *'try middle'* (spróbuj element środkowy).

Jeśli warunek *'first<=last'* nie jest spełniony *('f')*, mamy sytuację opisaną w kolumnie 4. Następuje przejście od stanu *'1:Start'* do stanu *'3:End'*. Odpowiadająca pozycja dla kolumny 4 w zbiorze {*TRANSITION*} to *'failure'* i w zasadzie nie ma czemu się dziwić. Przeszukiwanie w takiej sytuacji nie ma sensu. Każda kolumna infoMapy przedstawionej na rysunku 9 opisuje inny stan programu. InfoMapa 9 stanowi zestawienie możliwych stanów dwóch porównywanych programów.

Z powyższych przykładów wynika, że infoMapy mogą być wykorzystywane do porównywania oraz optymalizowania algorytmów. Możliwe jest także sprawdzenie czy dany 'nowy' nie jest w gruncie rzeczy tym samym, nieco zmienionym, już istniejącym wcześniej algorytmem oraz, czy rzekomo 'nowa' metoda rzeczywiście wnosi coś nowego do inżynierii oprogramowania.

Jako kolejny przykład opiszemy rozwój analizatora syntaktycznego oryginalnie dyskutowany w [14]. Załóżmy, że na pewnym poziomie tworzenia tego analizatora otrzymaliśmy już graf przepływu

	2	3	4	5	6	7	8	9	10	11	12	13	14
1													
2	A	A	A	A	A	A	A	A	A	A	A		{AUTHOR}
3	v	v	v	v	v	v	v	v	v	v	v		InfoSCHEMA Copyright © W.M. Jaworski
4	v	v	v	v	v	v	v	v	v	v	v		InfoMAP Copyright © WMJ
5	A	A	A	A	A	A	A	A	A	A	A		{VIEW's Purpose}
6	v												Pascal-like Code for Binary Search
7		v	v	v	v	v	v	v	v				Pascal-like Code rewritten as infoMAP (Version 1 and 2)
8										v			infoMap rewritten as Pascal-like Code
9											v		Cardinality
10	S									S	23		{STATEMENT}
11	1												Start: IF (first <= last) THEN i:= (first+last) div 2;
12	2												ELSE X not found; GOTO Exit;
13	3												IF a[i]<X THEN first := i-1; GOTO Start;
14	4												ELSE
15	5												IF a[i]>X THEN last:= i-1 GOTO Start;
16	6												ELSE X found at i GOTO Exit;
17	7												Exit:
18										1			1: IF (first <= last) THEN i:= (first+last)div 2 GOTO 2;
19										2			IF NOT(first <= last) THEN X not found GOTO 3;
20										3			2: IF a[i]<X THEN first := i+1 GOTO 1;
21										4			IF (NOT(a[i]<X)) AND (a[i]>X) THEN last:= i-1 GOTO 1;
22										5			IF (NOT(a[i]<X)) AND NOT(a[i]>X) THEN X found at i GOTO 3;
23										6			3: End;
24		A	A	A	A	A	A	A	A		2		{PROCESS}
25		v	v	v	v	v							Binary Search Program Version 1
26							v	v	v				Binary Search Program Version 2
27		O	O	O	O	O	O	O	O		5		{TRANSITION}
28		o											Try middle
29			o										failure
30				o			o						try upper
31					o			o					try lower
32						o			o				success
33		L	L	L	L	L	L	L	L		6		{STATE}
34		s	s	d	d								1: Start
35		d		s	s	s							2: Process element a[i]
36			d			d							3: End
37							1	1	s				1: Process element a[i]
38							a	a					2: Failure
39									d				3: Success
40		G	G	G	G	G	G	G	G		3		{preCONDITION}
41		t	f										first <= last
42				t	f	f	t	f	f				a[i]<X
43					t	f		t	f				a[i]>X
44		S	S	S	S	S	S	S	S		5		{ACTION}
45				1			1						first := i+1
46					1			1					last:= i-1
47		1					2	2					i:= (first+last) div 2;
48						1			1				X found at i
49			1										X not found
50		G	G	G	G	G	G	G	G		1		{postCONDITION}
51							f	f					first <= last
52		F	F	F	F	F	F	F	F		5		{DATA}
53				i	i	i	i	i	i				a[FIRST..LAST]::= vector
54				i	i	i	i	i	i				X::= searched for existance in vector a
55		i	i	o			b	i					first::= lower limit
56		i	i		o		i	b					last:;= upper limit
57		o		i	i	i	b	b	i				i::= middle
58													
59		Rys. 9. Zestawienie dwóch algorytmów przeszukiwania binarnego.											

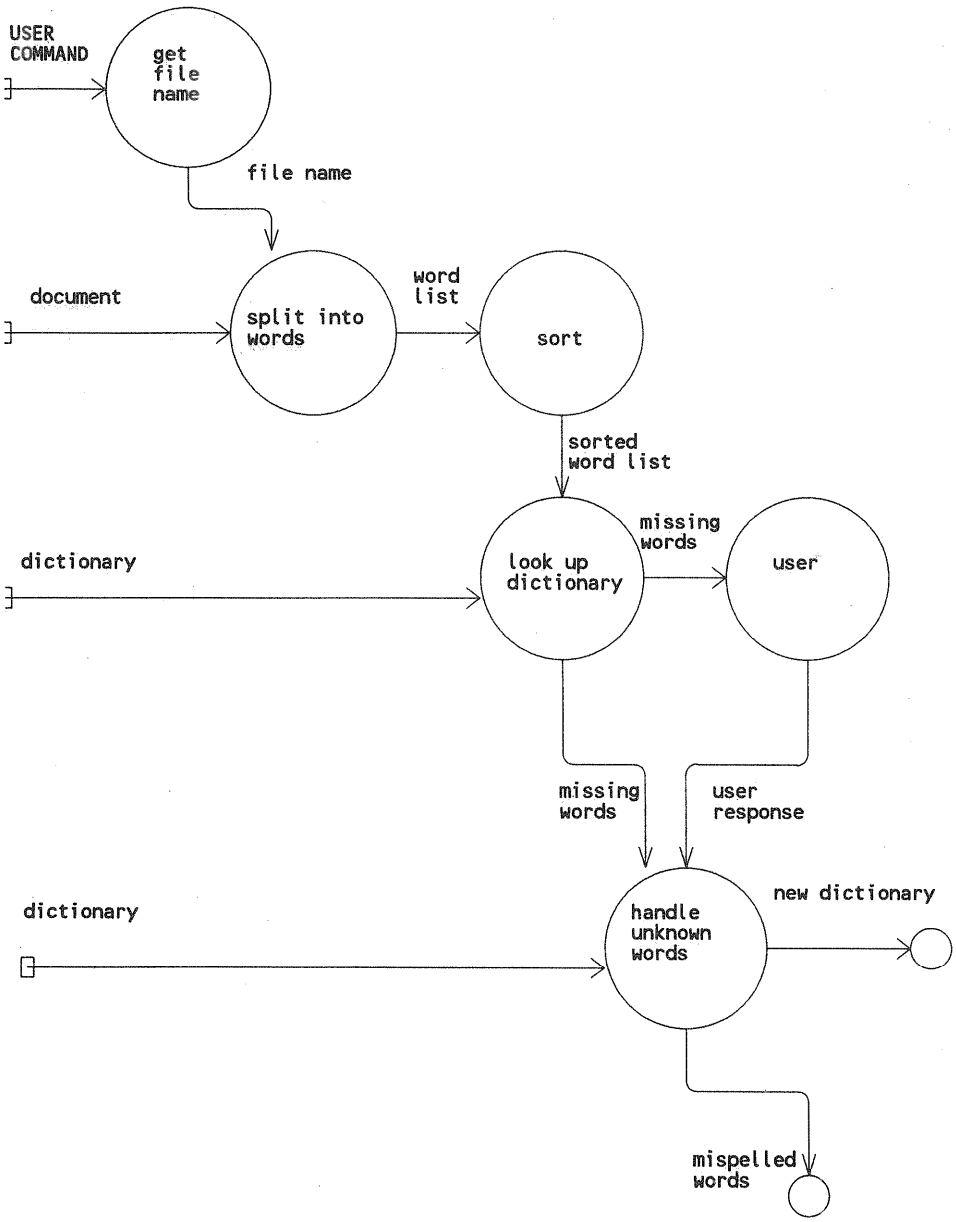


Rys. 10. infoSYNTAX

danych pokazany na rysunku 11 (będący adaptacją rysunku 18 z [18]). Yau daje zbiór relacji między tym grafem przepływu a diagramem struktury z rysunku 13. Graf przepływu i diagram struktury są wyrażone jako infoMapy na rysunkach 12 i 14. Przykładowo schemat rysunku 12(a) mówi nam o składnikach rysunku 12(b): Są to OPERATION i dataOBJECT. Górna część infoMapy z nagłówkiem złożonym z sześciu liter 'O' na rysunku (12b) definiuje sześć operacji odpowiadających sześciu wierzchołkom grafu przepływu danych[1].

Dolna część infoMapy opisuje dwie binarne relacje od operacji do obiektów. Wiersz zawierający *'g'* odpowiada obiektowi wejściowemu (lub inaczej danemu: *given*). Wiersz odpowiadający obiektowi wyjściowemu jest oznaczony symbolem *'r'* (*result*). Schemat infoMapy na rysunku 14 jest bogatszy niż ten z rysunku 12. W części górnej po lewej stronie użyto symbolu *'H'* do oznaczenia formy drzewiastej diagramu struktury, symbole *'g' i 'r'* mają takie samo znaczenie jak zdefiniowane wcześniej. Symbol *'b'* wskazuje, że dany obiekt jest zarówno wprowadzany jak i zwracany przez operację. Przykładowo *'GetWords'* otrzymuje nazwę pliku z *'GetFileName'* i wysyła ją do *'split'*. Użycie kilku różnych symboli w treści

[1] Graf skierowany $G<\{O\},\{D\}>$, może być określony jako zbiór wierzchołków oznaczony literą *'O'* i zbiór krawędzi oznaczony literą *'D'*. Do identyfikacji krawędzi zbioru G, używa się liter *'g','r'* oraz *'b'* (*given, result, both*)

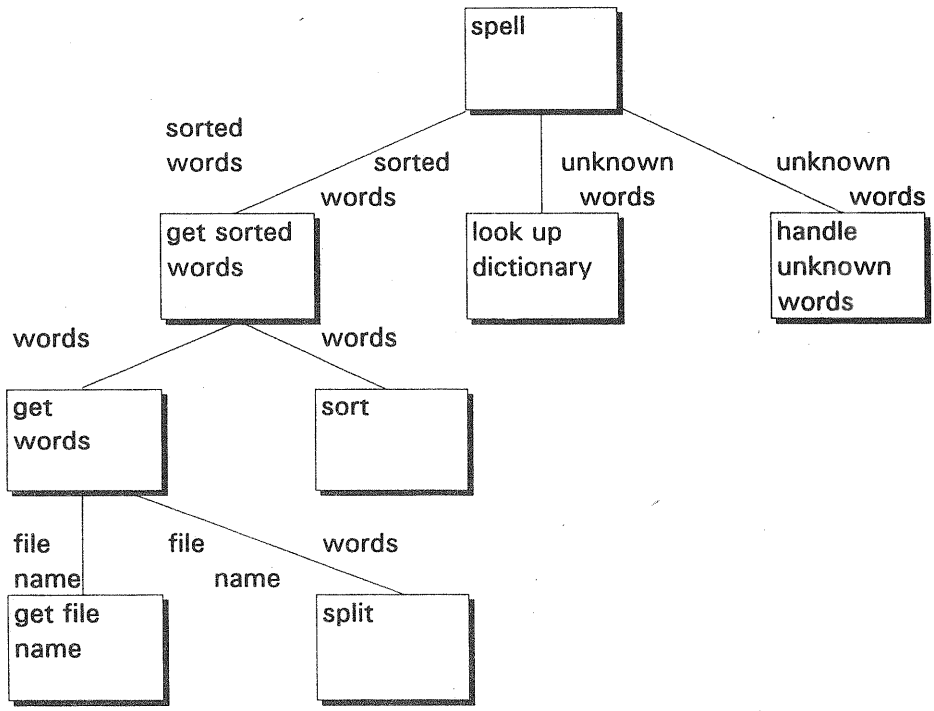


RYS. 11. Graf przepływu danych

jMapy odpowiada użyciu kilku rodzajów krawędzi w grafie.

Niejednoznaczności notacji można uniknąć, jeśli wybierze się symbole z wielkiego alfabetu (małe i duże litery, cyfry, litery greckie itd.). Liczba rodzajów krawędzi jaka może być rozróżniana (jasne, ciemne, przerywane itp.) jest ograniczona. Prezentacja grafów wymaga ponadto właściwego oprogramowania, gdyż trudno jest wyświetlić na ekranie graf mający wiele wierzchołków. Nieetykietowana krawędź grafu zawiera tylko jeden bit informacji; osobie oglądającej mówi tylko czy dwa węzły są w relacji. W celu zinterpretowania grafu trzeba dokładnie wiedzieć, co oznacza obecność krawędzi (kontekst).

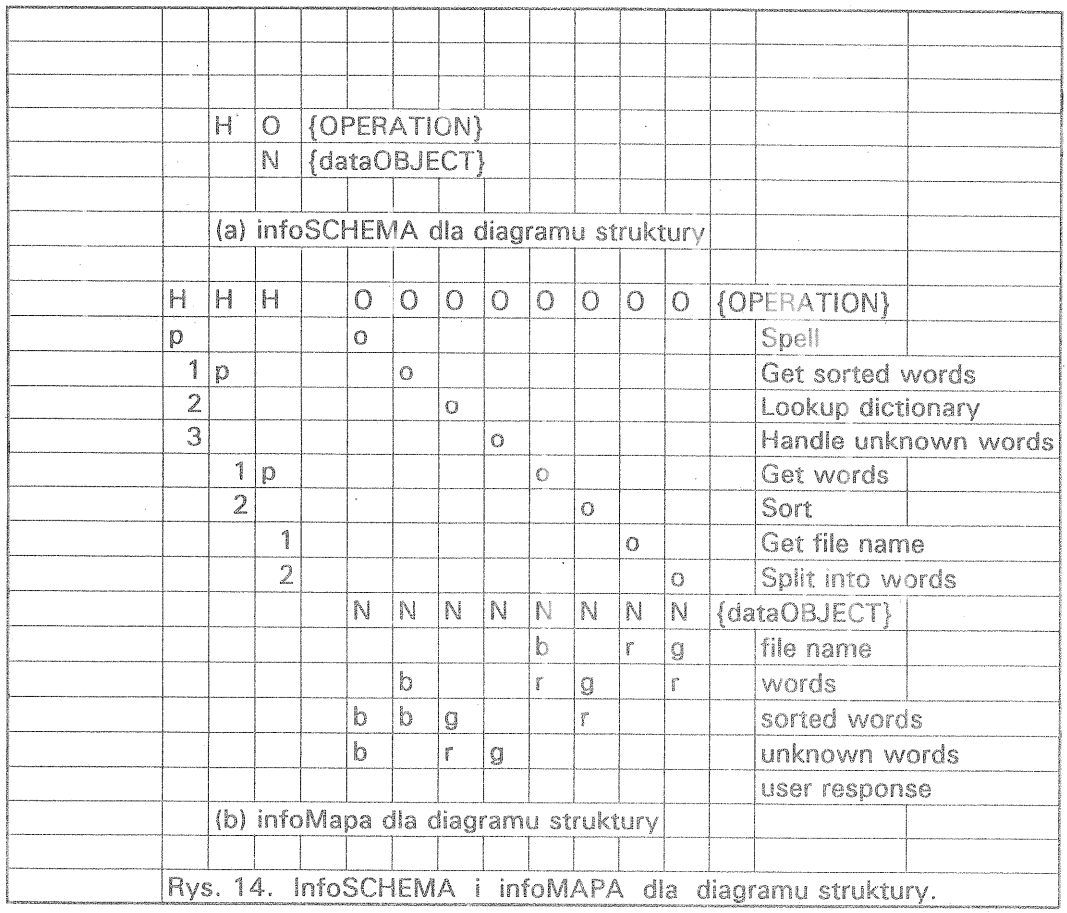
W grafie przepływu danych na rysunku 11 każda krawędź odpowiada przepływowi danych pewnego rodzaju. Przykładowo, krawędź *'sorted word list'* łączy słowa z węzła *'sort'* z węzłem *'lookup dictionary'*. Na diagramie struktury (rys. 13) węzły są połączone, jeśli jest jakikolwiek przepływ danych między nimi.



Rysunek 13. Diagram struktury dla grafu z rysunku 11.

		O	{OPERATION}								
		N	{dataOBJECT}								
	(a) infoSCHEMA dla grafu przepływu danych										
	O	O	O	O	O	O	{OPERATION}				
	o							lookup dictionary			
		o						handle unknown words			
			o					sort			
				o				get file name			
					o			split into words			
						o		user			
	N	N	N	N	N	N	{dataOBJECT}				
	g	g						dictionary			
		r						new dictionary			
					g			document			
				g				user command			
		r						mispelled words			
				r	g			file name			
			g		r			word list			
	g		r					sorted word list			
	r	g				g		missing words			
		g				r		user response			
	(b) infoMapa dla grafu przepływu danych										

Rys. 12. infoSCHEMA i infoMAPA dla diagramu przepływu danych.



Rys. 14. InfoSCHEMA i infoMAPA dla diagramu struktury.

Strzałki wskazują zależność a nie kierunek przepływu. Przykładowo: węzeł *'lookup dictionary'* otrzymuje posortowane słowa a zwraca słowa nieznane.

W reprezenatacji jMapy takie niewidoczne rozróżnienia nie istnieją. Znaczenie każdej pozycji w jMapie jest jasne dzięki zawartości wiersza deskryptorów. W prezentowanej konwencji możemy używać 'krawędzi' różnych rodzajów bez niejednoznaczności.

4. Wnioski

Omówione przykłady pozwoliły - mamy nadzieję - skutecznie zaprezentować użyteczność InfoMap jako narzędzia do analizy konstrukcji stosowanych w językach programowania i algorytmach. Te możliwości odnoszą się także do specyfikacji oraz konstrukcji projektowych. Notacja infoMap jest oparta o relacje, grafy i funkcje. Stanowi doskonałe narzędzie do dokumentowania, zmniejszania redundancji poprzez wykrywanie ukrytych powiązań. Pozwala również na modelowanie różnych obiektów z uwzględnieniem ich aspektów statycznych i dynamicznych. InfoMapy stanowią także poważny krok w kierunku utworzenia 'Przestrzeni Informacyjnej Systemu' [15], wspierającej projektantów systemu we wszystkich fazach rozwojowych tego systemu. Propozycja implementacji tej idei jest przedstawiona w pracy [17] jako 'przestrzeń informacyjna projektu.'

InfoMapy pozwalają na bardzo łatwe przetwarzanie zawartych w nich informacji np. przez użycie typowych arkuszy kalkulacyjnych oraz narzędzi rozszerzających te arkusze [5].
Narzędzia te pozwalają na generację kodu programu na podstawie specyfikacji programu w postaci infoMapy. I żeby zakończyć prezentację możliwości infoMap możemy powiedzieć, iż stanowią one narzędzie pozyskiwania i gromadzenia

wiedzy o możliwościach nie w pełni jeszcze odkrytych. Jeszcze raz namawiamy Czytelnika na zmierzenie się z rysunkiem 3, na którym jest przedstawiona w sposób ustrukturyzowany wiedza z dziedziny bardzo odległej od przykładów ilustrujących niniejszą prezentację.

Bibliografia

[1] Aho A. V., Hopcroft J. E., Ullman J. D.: *Projektowanie i analiza algorytmów komputerowych*, PWN, Warszawa 1983.

[2] Amstel J. J., Poirters J. A. A. M.: *The design of Data Structures and Algorithms*, Prentice Hall, Academic Service, 1989.

[3] Berztiss A. T.: *Data Structures. Theory and Practice*, Academic Press. New York, London 1971.

[4] Dijkstra E. W.: *Guarded commands, nondeterminancy and the formal derivation of programs*, CACM 18, August 1975, pp. 453-457.

[5] *InfoFARM: User Manual*, GS General Strategies Inc., 1991.

[6] Jaworski W. M., Ficocelli L., O'Mara K. S.: *The ABL/W4 Methodology for System Modelling*, System Research J., 1987, Vol. 4, No. 1, pp. 23-27.

[7] Jaworski W. M., Virard M.: *Converting a Software Company to a New Technology*, Proceedings of the 1986 Canadian Conference on Industrial Computers, Montreal 1986, pp 12-1..12-7.

[8] Jaworski W. M., MacCuaig I., Marinelli T., Nysztor T.: *Executable Specification for a Large Industrial Process*, Proceedings of the 1986 Canadian Conference on Industrial Computers, Montreal 1986, pp. 60-1..60-5.

[9] Jaworski W. M., Cummings T.: *Program Normalization and Optimization: Using infoMaps as an inspection and program processing tool*, Canadian Conference on Electrical and Computer Engineering, Quebec City, September 1991.

[10] Jaworski W. M.: *System Analysis and Design in a Classroom: infoMAPs Teaching Factory*, Modelling Conference, Pittsburg, Pa., May 2-3, 1990.

[11] Jaworski W. M., Deslauriers B.: *Software Process, Maintenance, Products: Software Development Environment based on infoMAPs*, Control and Electrical Eng. Conference, Ottawa, September 1990.

[12] Jaworski W. M., Grogono P. D.: *infoMAPS: a Pragmatic Environment for Seamless and Nondeterministic Software Development*, Concordia University, Computer Science Dept., January 1991.

[13] Kovats T. A.: *Comments on innovative control constructs in Pancode and EPN*, SIGPLAN Notices 25, 11 (November 1990).

[14] Somerville I.: *Software Engineering*, Addison-Wesley, 1986.

[15] Webster D.: *Mapping the design representation terrain: a survey*, IEEE Computer, (December 1988), pp. 8-23.

[16] Wirth N.: *Algorytmy+Struktury danych=Programy*, WNT, Warszawa 1989.

[17] Zaliwski A., Kuraś M.: *CASAD - Nowe podejście do modernizacji SI*, Materiały Piątej Jubileuszowej Wyższej Międzynarodowej Górskiej Szkoły PTI, Szczyrk 1993.

[18] Yau S., Nicholl R., Tsai J. and Liu S. S.: *An integrated life cycle for software maintenance*, IEEE Trans. Soft. Eng., 1988, 14, 8, 1128-44.