Volume 16 Number 1 March 1992
YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

The Slovene Society INFORMATIKA
Ljubljana

Informatica

A Journal of Computing and Informatics

Subscription Information

Informatica (YU ISSN 0350 – 5596) is published four times a year in Winter, Spring, Summer and Autumn (4 issues).

The subscription price for 1992 (Volume 16) is US$ 30 for companies and US$ 15 for individuals.

Claims for missing issues will be honoured free of charge within six months after the publication date of the issue.

Printed by Tiskarna Tomi Pretnar, Bratov Komel 52, Ljubljana.

Informacija za naročnike

Informatica (YU ISSN 0350 – 5596) izide štirikrat na leto, in sicer v začetku februarja, maja, avgusta in novembra.

Letna naročnina v letu 1992 (letnik 16) se oblikuje z upoštevanjem tečaja domače valute in znaša okvirno za podjetja DEM 30, za zasebnike DEM 15, za študente DEM 8, za posamezno številko pa DEM 10.

Številka žiro računa: 50101 – 678 – 51841.

Zahteva za izgubljeno številko časopisa se upošteva v roku šestih mesecev od izida in je brezplačna.

Tisk: Tiskarna Tomi Pretnar, Bratov Komel 52, Ljubljana.

Na podlagi mnenja Ministrstva za informiranje št. 23/216-92, z dne 27.3.1992, šteje znanstveni časopis Informatica med proizvode informativnega značaja iz točke 13 tarifne številke 3, za katere se plačuje davek od prometa proizvodov po stopnji 5%.

Pri financiranju časopisa **Informatica** *sodeluje Ministrstvo za znanost in tehnologijo, Slovenska 50, 61000 Ljubljana*

Volume 16 Number 1 March 1992
YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

EDITOR – IN – CHIEF
Anton P. Železnikar
Volaričeva ulica 8, 61111 Ljubljana

ASSOCIATE EDITOR
Rudolf Murn
Jožef Stefan Institute, Ljubljana

The Slovene Society INFORMATIKA
Ljubljana

Letnik 16 Številka 1 Marec 1992 YU ISSN 0350 – 5596

Informatica

Časopis za računalništvo in informatiko

V S E B I N A

NAIVE BAYESIAN CLASSIFIER AND CONTINUOUS ATTRIBUTES

INFORMATICA 1/92

Keywords: machine learning, naive Bayesian classifier, continuous atributes
Ključne besede: avtomatsko učenje, naivni Bayesov klasifikator, zvezni atributi

Igor Kononenko
University of Ljubljana
Faculty of Electrical and Computer Engineering
Tržaška 25, 61001 Ljubljana, Slovenia

Abstract

The advantages of the naive Bayesian classifier are fast and incremental learning, robustness with respect to missing data, the inclusion of all available attributes during classification and the explanation of classification as the sum of information gains. Besides the 'naivety', the weakness is also the inability to deal with continuous attributes unless they are discretized in advance. In the paper three methods for dealing with continuous attributes are proposed. The fuzzy learning method assumes fuzzy bounds of a continuous attribute during learning, the fuzzy classification method assumes fuzzy bounds during classification and the last method tries to increase the reliability of probability approximations. The performance was tested in two medical diagnostic problems.

Naivni Bayesov klasifikator in zvezni atributi
(povzetek)

Prednosti naivnega Bayesovega klasifikatorja so hitro in inkrementalno učenje, robustnost glede manjkajočih podatkov, uporaba vseh razpoložljivih atributov za klasifikacijo in razlaga odločitev kot vsota informacijskih prispevkov. Poleg naivnosti je slabost tudi nezmožnost obravnave zveznih atributov, ki morajo biti zato vnaprej diskretizirani. V članku so predstavljene tri metode za obravnavanje zveznih atributov. Mehko učenje in mehko testiranje predpostavljata mehke meje zveznih atributov med učenjem oziroma med klasifikacijo. Tretja metoda temelji na povečanju zanesljivosti aproksimacij verjetnosti. Uspešnost je bil testirana na treh medicinskih diagnostičnih problemih.

1 Introduction

The basic Bayesian formula for calculating the probability of a class given the values of attributes, describing a given object, can be used either for generation of decision trees (Michie & AlAttar 1990) or can be simplified into 'naive' Bayesian classifier by assuming the independence of attributes (Kononenko 1989).

The advantages of the naive Bayesian classifier are fast and incremental learning, robustness with respect to missing values, the inclusion of all available attributes for classification, and the ability to explain decisions as the sum of information gains (Kononenko 1990).

It was shown by several authors that, despite 'naivety', the naive Bayesian classifier performs in many real world problems approximately the same or even better than many well known inductive learning systems (Bratko & Kononenko 1987, Cestnik 1990, Kononenko 1990).

The advantages of induction of decision trees are 'nonnaivety', *simple and powerful mechanism for on line splitting of continuous attributes*, and explicit knowledge in the form of if-then rules. This paper is concerned with the problem of dealing with continuous attributes. The problem is to split the interval of possible values of a continuous attribute into subintervals to obtain as much as possible useful information for classification from a given attribute.

In the algorithms for induction of decision trees a continuous attribute is binarized on-line during learning. In a given node of a tree a bound is selected that maximizes the information gain of the attribute (Breiman et al. 1984, Paterson & Niblett 1982, Bratko & Kononenko 1987, Quinlan 1986). This approach can be applied for the naive Bayesian classifier only at the highest level by changing all continuous attributes into binary attributes. Another approach is to define all subintervals of-line before learning, either by a human expert or with an algorithm (e.g. Cestnik 1989). However, all these approaches assume, that the optimal split can be obtained using *exact bounds* of subintervals. This is unrealistic assumption as typically the slight difference among two values, one above and one below the bound, should not have drastic effect.

In this paper three new methods for dealing with continuous attributes are proposed. Two of them are based on the idea of *fuzzy bounds* and the third one is based on the reliability of probability estimation. In next section the naive Bayesian classifier is briefly described. In section 3 the three methods for dealing with continuous attributes are defined and in section 4 the experiments in two medical diagnostic problems are described. Finally in section 5 some conclusions are given and the further research is proposed.

2 Naive Bayesian Classifier

The classification problem discussed in this paper is the following: given a set of training instances, each described with a set of n attributes and each belonging to exactly one of a certain number of possible classes, learn to classify new, unseen objects. In addition, each attribute A_i has a fixed number of NV_i possible values.

Let C represent one of the possible classes. Let V_{i,J_i} be a Boolean variable having value 1 if the current instance has value J_i of i-th attribute and 0 otherwise. The conditional probability of class C given the values of all attributes is given with the following formula, derived from the Bayesian rule (Good 1950) (for brevity conditions $V_{i,j}=1$ will be written simply as $V_{i,j}$):

$$P(C|V_{1,J_1},\ldots,V_{n,J_n}) = P(C)\prod_{i=1}^{n} Q_i(C,J_i) \tag{1}$$

where

$$Q_i(C,J_i) = \frac{P(V_{i,J_i}|C,V_{1,J_1},\ldots,V_{i-1,J_{i-1}})}{P(V_{i,J_i}|V_{1,J_1},\ldots,V_{i-1,J_{i-1}})}$$

$$= \frac{P(C|V_{1,J_1},\ldots,V_{i,J_i})}{P(C|V_{1,J_1},\ldots,V_{i-1,J_{i-1}})} \tag{2}$$

and $P(C)$ is the prior probability of class C. It was shown in (Kononenko 1989) that the classification with ID3 like inductive learning system can be described with (1).

From (1) the naive Bayesian classifier, as used by Bratko and Kononenko (1987), is obtained if the independence of attributes is assumed. Eq. (1) remains unchanged except that factors Q_i defined with (2) are replaced with Q_i' (we will refer to changed equation (1) with (1')):

$$Q_i'(C,J_i) = \frac{P(V_{i,J_i}|C)}{P(V_{i,J_i})} = \frac{P(C|V_{i,J_i})}{P(C)} \tag{3}$$

The probabilities necessary to calculate (3) are approximated with relative frequencies from the training set. A new object is classified by

calculating the probability for each class using equation (1') and the object is classified into the class that maximizes the calculated probability.

Cestnik (1990) has shown that the kind of approximation of probabilities in (3) considerably influences the classification accuracy of the naive Bayesian classifier. Let $N(C, V_{i,J_i})$ be the number of training instances with J_i-th value of i-th attribute and belonging to class C and let $N(V_{i,J_i})$ be the number of training instances with J_i-th value of i-th attribute. Usually, the probability is approximated with relative frequency, i.e.

$$\hat{P}(C|V_{i,J_i}) = \frac{N(C, V_{i,J_i})}{N(V_{i,J_i})} \qquad (4)$$

However, if the training set is relatively small, the corrections are needed with respect to the assumption of initial distribution (Good 1965). Cestnik (1990) used the following formula stemming from the assumption, that initial distribution of classes is equal to $P(C)$:

$$\hat{P}(C|V_{i,J_i}) = \frac{N(C, V_{i,J_i}) + 2\hat{P}(C)}{N(V_{i,J_i}) + 2} \qquad (5)$$

where the probability of class C is calculated using the Laplace's law of succession (Good 1950,1965):

$$\hat{P}(C) = \frac{N(C) + 1}{N + 2} \qquad (6)$$

Cestnik has shown some nice properties of using approximation (5) in formula (3) and has shown experimentally, that the naive Bayesian classifier using approximation (5) performs significantly better than if (4) is used. The same formula was used also by Smyth and Goodman (1990).

3 Dealing with continuous attributes

The idea of all three methods defined in this section is the folowing. The task is to calculate the probabilities of all classes of an object with a given value of a continuous attribute. These probabilities should be approximated with relative frequencies calculated from the distribution of training instances with the similar value of the attribute. It is assumed, that small variations of the value of the attribute should have small effects on the probabilities. As opposed to exact bounds, where slightly different value can have drastic effects on the calculated probabilities, the bounds of intervals are here assumed to be *fuzzy*.

The three methods described in this section differ in the way how the distribution, used in the approximation of probabilities, is obtained. For all three methods the pessimistic set of possible bounds is given in advance either by a human expert or with a simple algorithm, that returns bounds with the uniform distribution of instances over all intervals. The set of bounds is pessimistic in the sense that more bounds are given than probably needed (e.g. all attributes have in advance 20 possible intervals, which is typically too detailed split). However, exact values of these initial bounds are not important and may vary without significant changes in performance.

3.1 Fuzzy learning

The fuzzy learning is performed by calculating the probability distribution for a given interval from all training instances rather than from instances that have value of a given continuous attribute in this interval. The influence of an instance is assumed to be normally distributed with mean value equal to the value of the regarded attribute and with given σ. σ is the parameter to the learning algorithm and is used to control the 'fuzziness' of the bounds. As shown in figure 1, the influence of a given instance with value v of the given continuous attribute on the distribution of interval $(b_j..b_{j+1})$ is proportional to the following expression:

$$P(v, \sigma, j) = \int_{b_j}^{b_{j+1}} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-v}{\sigma}\right)^2} dx \qquad (7)$$

If $\sigma = 0$ then the usual exact bounds are assumed and the distribution over classes in the

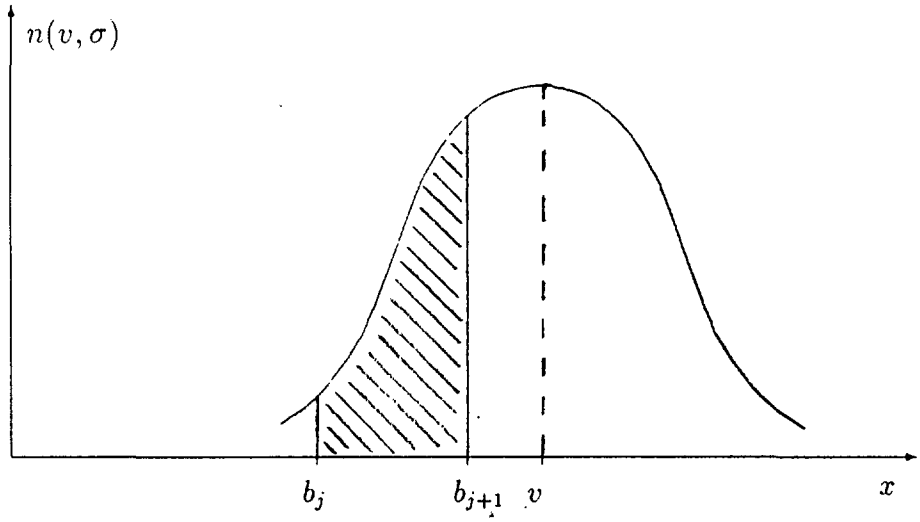


Figure 1: The normal distribution of the influence of:
- an instance over intervals of a continuous attribute for fuzzy learning,
- an interval on classification for fuzzy testing

given interval is calculated from relative frequency of training instances belonging exactly to that interval. The greater σ implies fuzzier bounds for continuous attributes. For example, for N learning instances, each with influence $P(v_k, \sigma, J_i)$, $k = 1..n$ on J_i-th interval of i-th attribute the probability that an instance belongs to that interval is calculated with:

$$\hat{P}(V_{i,J_i}) = \frac{\sum_{k=1}^{N} P(v_k, \sigma, J_i)}{N} \qquad (8)$$

3.2 Fuzzy classification

The fuzzy classification is performed by calculating the probability of all classes for a given object, given the value of the continuous attribute, from all intervals of that attribute rather than from the interval to which the object belongs. The influence of intervals is assumed to be normally distributed with mean value equal to the value of the regarded attribute of an object and with given σ.

σ is like in the previous method the parameter to the classification algorithm and is used to control the 'fuzziness' of the bounds. As shown on figure 1, the influence of a given interval on the probability of all classes is proportional to the expression (7). The expression (3) in equation (1') is here replaced with:

$$Q_i"(C) = \sum_{j=1}^{NV_i} P(v, \sigma, j) \times \frac{\hat{P}(C|V_{i,j})}{\hat{P}(C)} \qquad (9)$$

where v is the value of i-th attribute of a given object. Like in fuzzy learning method, for $\sigma = 0$ the usual exact bounds are assumed and the probabilities of all classes for the given object are calculated from relative frequencies of one interval only.

3.3 Reliable approximations of probabilities

The idea of the last method is to increase the reliability of approximation of the probabilities for given interval by adding to the interval the instances from neighbor intervals. For the estimation of the reliability of the probability approximation the theorem of Chebyshev (e.g. Vadnal 1979) can be used. The theorem gives the lower bound on the probability, that relative frequency f of an event after n trials differs from the factual prior probability p for less than ε:

$$P(|f - p| \leq \varepsilon) > 1 - \frac{p(1-p)}{\varepsilon^2 n} \qquad (10)$$

The lower bound is proportional to n and to

Table 1: Characteristics of two medical data sets.

domain	thyroid	rheumatology
# instances	884	355
# classes	4	6
# attributes	15	32
average # vals/att	9.1	9.1
average #missing data	2.7	0.0
majority class	56%	66%
entropy (bit)	1.59	1.70
# continuous atts	7	22
average # intervals/att	17.3	10.0

ε^2. In our case we are interested in the reliability of the approximation of probability (5). Therefore the number of trials n in (10) is equal to $N_{V_{i,J_i}}$, i.e. the number of training instances having value inside interval V_{i,J_i} of attribute A_i. As prior probability p is unknown, in our experiments for approximation of p at the right-hand side of (10) the worst case was assumed, i.e. $p = 0.5$. It remains to determine the value of ε. The influence of interval J_i of i-th attribute to class C is proportional to the difference between (5) and (6). If the influence is greater the reliability of approximation of (5) should be greater. Therefore ε should be proportional to the influence. As we regard the influence of an interval for all the classes C_j, $j = 1..n$, for ε the average difference is used:

$$\varepsilon = \sum_{j=1}^{n} \hat{P}(C_j) \times |\hat{P}(C_j|V_{i,J_i}) - \hat{P}(C_j)| \quad (11)$$

From above formulas it follows that the interval is unreliable if:

$$1 - \frac{1}{4\varepsilon^2 N_{V_{i,k}}} < P_r \quad (12)$$

where P_r is the lower bound of the probability (10) and is a parameter of the learning algorithm for controlling the reliability of approximations of probabilities. Now we can define an algorithm for postprocessing the distributions obtained by usual learning for naive Bayesian classifier. The algorithm is as follows:

for each continuous attribute A_i **do**
 for each interval $V_{i,j}$ **do**
  while unreliable interval $V_{i,j}$ **do**
   add appropriate % (possibly 100%)
    of instances from neighbor
    intervals of attribute A_i

Note that here one training instance can influence more than one interval of a continuous attribute, analogously to fuzzy learning method.

4 Experiments in two medical diagnostic problems

We experimented with the naive Bayesian classifier and with three methods for dealing with continuous attributes in two medical diagnostic problems: diagnosing thyroid diseases and rheumatology. The characteristics of data sets used in our experiments are summarized in table 1. The medical data sets were collected at the University Medical Center in Ljubljana.

One run was performed by randomly selecting 70% of instances for learning and 30% for testing. Results are averages of 10 runs. For fuzzy learning and fuzzy testing methods parameter σ was determined with the following formula for each continuous attribute A_i:

$$\sigma_i = SIG \times \frac{upperbound_i - lowerbound_i}{\#intervals_i}$$

Table 2: Results of fuzzy learning and fuzzy classification.

method	SIG	thyroid acc(%)	thyroid inf.(bit)	rheumatology acc(%)	rheumatology inf.(bit)
-	0.0	69.7	0.79	67.4	0.51
f.learn.	0.3	71.7	0.84	67.9	0.54
f.learn.	0.4	71.9	0.85	67.8	0.53
f.learn.	0.5	72.1	0.85	68.1	0.54
f.learn.	0.6	72.3	0.85	68.1	0.54
f.learn.	0.7	72.5	0.86	68.1	0.54
f.learn.	0.8	72.5	0.85	68.0	0.54
f.class.	0.3	71.4	0.84	59.6	0.38
f.class.	0.4	71.6	0.84	60.7	0.40
f.class.	0.5	70.8	0.82	62.6	0.45
f.class.	0.6	70.8	0.82	62.0	0.43
both	0.3	72.2	0.85	68.9	0.58
both	0.4	72.2	0.85	68.7	0.57
both	0.5	72.3	0.86	69.0	0.58
both	0.6	72.2	0.85	69.4	0.59

Table 3: Results with reliable approximations of probabilities (for $P_r = 0.0$ the result is of the original naive Bayesian classifier without reliable approximation of probabilities).

P_r	thyroid acc(%)	thyroid inf.(bit)	rheumatology acc(%)	rheumatology inf.(bit)
0.0	69.7	0.79	67.4	0.51
0.1	70.6	0.80	67.0	0.51
0.2	71.0	0.80	67.5	0.52
0.3	70.9	0.80	67.9	0.53
0.4	70.9	0.80	68.3	0.53
0.5	70.7	0.79	69.2	0.55
0.6	70.5	0.79	69.3	0.53

Table 4: The comparison of performance of different classifiers in two medical domains.

classifier	thyroid		rheumatology	
	acc(%)	inf(bit)	acc(%)	inf(bit)
f.learn.	73	0.86	68	0.54
f.class.	72	0.84	61	0.40
f.learn&class	72	0.86	69	0.58
reliable app.	71	0.80	69	0.55
naive Bayes	70	0.79	67	0.51
Assistant	73	0.87	61	0.46
physicians	64	0.59	56	0.26

where SIG is parameter that was varied in our experiments. If $SIG = 1$ then σ_i is equal to the average interval length for i-th attribute. Therefore fuzziness is the function of average length of intervals. For last method the parameter P_r from (12) was varied.

Besides the average percent of correct guesses, the average information score per answer (Kononenko & Bratko 1991) was measured. Information score is the measure that eliminates the influence of prior probabilities of classes and can be applied to various kinds of incomplete and probabilistic answers. This measure is necessary as in each domain a classifier that classified each instance into the majority class (see table 1) would achieve high classification accuracy.

Results are given in tables 2 and 3. In table 2, the results are given for fuzzy learning, fuzzy classification and both methods together for various values of parameter SIG. In table 3 the results of reliable approximations of probabilities are presented for various values of parameter P_r. The combination of the latter method with other two did not give any improvements. In table 4 the results are compared with the accuracy of ID3-like inductive learning system Assistant (Cestnik et al. 1987), the naive Bayesian classifier without any method for dealing with continuous attributes and with the performance of physicians experts. The performances of physicians are the averages of four physicians experts in each domain that were tested in University Medical Center in Ljubljana.

5 Discussion

The results from table 4 show that proposed methods for dealing with continuous attributes perform better than splitting of attribute's values with exact bounds. The performance of the naive Bayesian classifier with these methods achieves and outperforms the performance of Assistant inductive learning system as well as the performance of physicians specialists. The performance of physicians is the worse, probably due to the inability to see the patient during the diagnostic process, when they were tested. Such diagnosing is, of course, unusual and unnatural for physicians. The results are presented to show that the performance of the learning systems is high enough, and not to show that the systems are better than physicians.

All three proposed methods for dealing with continuous attributes are based on the idea that a continuous attribute should not be discretized with exact bounds. It is interesting that none of the methods uses the information gain of the attribute as the measure for appropriate split. Further research should concentrate on the selection of appropriate values of parameters P_r and SIG. Obviously, optimal value of two parameters may differ among different attributes in the same problem domain. The appropriate value of the parameter may depend on the information gain of the attribute as well as on the amount of noise associated with values of the attribute.

The problem with the naive Bayesian classifier is the independence assumption. In some cases this may be too unrealistic assumption. But it seems that in the data used by human experts there are no strong dependencies between attributes because attributes are properly defined. With the independence assumption the reliability of approximating factors with relative frequencies is much greater. This is supported with experimental results. The naive Bayesian classifier despite its naiveness achieved good classification accuracy. There is a trade-off between the reliability of approximating probabilities and the errors due to the independence assumption (Kononenko 1989). An algorithm that tries to optimize this trade-off is described elsewhere (Kononenko 1991).

Acknowledgements

Physicians specialists Sergej Hojker and Vlado Pirnat from University Medical Center in Ljubljana provided the medical data and tested the physicians' performance. This research was done in the Artificial Intelligence Laboratory at the Faculty of Electrical & Computer Engineering in Ljubljana and was supported by Slovenian Research Community.

References

I.Bratko & I.Kononenko, (1987) Learning Rules from Incomplete and Noisy Data, in B. Phelps (ed.) *Interactions in Artificial Intelligence and Statistical Methods*, Hampshire: Technical Press.

Breiman, L. Friedman, J.H., Olshen, R.A., Stone C.J. (1984), *Classification and Regression Trees*, Belmont, California: Wadsworth Int. Group.

Cestnik B., Kononenko I.& Bratko I., (1987) ASSISTANT 86 : A knowledge elicitation tool for sophisticated users, in: I.Bratko, N.Lavrac (eds.): *Progress in Machine learning*, Wilmslow, England: Sigma Press.

Cestnik B. (1989) Informativity based splitting of numerical attributes into intervals, *Proc. IASTED Intern. Conf. Expert Systems Theory & Applications*, Zurich, Switzerland, June 26-28, 1989, pp.59-62.

Cestnik B., (1990) Estimating Probabilities: A Crucial Task in Machine Learning, *Proc. European Conf. on Artificial Intelligence*, Stockholm, Sweden, August 1990, pp.147-149.

Good, I.J., (1950) *Probability and the Weighing of Evidence*, London: Charles Griffin.

Good I.J., (1965) *The Estimation of Probabilities*, Cambridge: M.I.T. Press.

Kononenko I., (1989) ID3, Sequential Bayes, Naive Bayes and Bayesian Neural Networks. *Proc. 4th European Working Session on Learning*, Montpeiller, France, 4-6 December 1989, pp.91-98.

Kononenko I. (1990) Comparison of inductive and naive Bayesian learning approaches to automatic knowledge acquisition, in: B.Wielinga, J.Boose, B.Gaines, G.Schreiber, M. van Someren (eds.) *Current Trends in Knowledge Acquisition*, Amsterdam: IOS Press.

Kononenko I. (1991) Semi-Naive Bayesian Classifier, *Proc. 5th European Working Session on Learning*, Porto, Portugal, March 1991, pp.206-219.

Kononenko, I. & Bratko, I., (1991) Information Based Evaluation Criterion for Classifier's Performance. *Machine Learning*, Vol. 6, pp.67-80.

Michie, D., A. Al Attar (1991) Use of Sequential Bayes with Class Probability Trees. in: J.E. Hayes-Michie, D.Michie & E. Tyugu (eds.) *Machine Intelligence 12*, Oxford: Oxford University Press.

Quinlan, J.R. (1986) Induction of Decision Trees. *Machine Learning.* Vol. 1, no. 1, pp. 81-106.

Paterson, A., Niblett, T. (1982) The ACLS User Manual. Intelligent Terminals Ltd. Glasgow.

Smyth P. & Goodman R.M. (1990) Rule Induction Using Information Theory in: G.Piarersky & W.Frawley (eds.) *Knowledge Discovery in Databases*, MIT Press.

Vadnal A. (1979) *An Elementary Introduction to Probability Calculus*, (in Slovenian), Ljubljana: Državna Založba Slovenije.

AN INFORMATIONAL APPROACH OF BEING-THERE AS UNDERSTANDING I*

INFORMATICA 1/92

Keywords: Heideggerian Being-there, information, informational formulas, philosophy, text formalization, understanding

Anton P. Železnikar
Volaričeva ulica 8
61111 Ljubljana

This essay is a study of possibilities pertaining to philosophical text formalization; it is a trial formalizing the most complicated and semantically interweaved concepts of Heidegger's Being-there as understanding. »What kind of formal informational system understanding in the context of Being-there could be?« is another, yet unanswered question. However, this essay is a beginning in formalizing the question of understanding in the context of Being (Dasein, Being-in, Being-in-the-world, Being-possible, Being-there, disclosure, existing, explaining, knowing, seeing, etc.). It is an approach towards the so-called informational understanding machine through the arising formalism (informational language).

The first part of the essay deals with introductory commentaries concerning the way of author's motivation to undertake such particular seeing of the problem. In the next section of the essay a formal informational interpretation of Being-there as understanding is presented in both Heidegger's and informationally formal way, in projecting and constructing formula systems which pertain to the original philosophical sentences. In this manner, in this part of the essay, fifteen paragraphs of Section 31 of Heidegger's *Being and Time* are interpreted in the informationally formal way. Several subscripted Greek and Fraktur letter operand and special operator symbols are introduced to make formulas readable and symbolically distinguishable.

In the continuation of the essay four further paragraphs of Heidegger's text will be formalized and an integrative formal interpretation of the examined paragraphs will be given. Two dictionaries of formal symbols together with explanation in English, German, and Slovene will be attached.

Informacijski pristop k biti-tu kot razumevanju I*

Ta spis je raziskava možnosti, ki se tičejo formalizacije filozofskih besedil; je poskus formaliziranja najbolj zapletenih in semantično prepletenih konceptov Heideggrove biti-tu kot razumevanja. Drugo, doslej neodgovorjeno vprašanje je, »Kaj je lahko razumevanje kot način formalnega informacijskega sistema v kontekstu biti (tubiti, biti-v, biti-v-svetu, biti-mogoče, biti-tu, razprtja, eksistiranja, pojasnjevanja, védenja, videnja in temu podobnega). Gre za poskus približevanja t.i. informacijskemu stroju razumevanja z uporabo nastajajočega formalizma (informacijskega jezika).

V prvem delu spisa najdemo uvodne komentarje, ki zadevajo napotovanje avtorjeve motivacije v táko posebno gledanje na problem. V naslednjem poglavju spisa je predstavljena formalna informacijska interpretacija biti-tu kot razumevanja, in sicer v Heideggrovi in informacijsko formalni obliki, v projektiranju in konstruiranju formulskih sistemov, ki zadevajo izvirne filozofske stavke. Tako je v tem delu spisa interpretiranih petnajst odstavkov 31. poglavja Heideggrovega dela *Bit in čas* v informacijsko formalni obliki. Vpeljani so različni operandni in operatorski simboli v obliki indeksiranih grških in gotskih črk, ki zagotavljajo bralnost in simbolno razločljivost formul.

V nadaljevanju spisa bodo formalizirani še štirje odstavki Heideggrovega besedila, dana pa bo tudi integralna formalna interpretacija obravnavanih odstavkov. Dva slovarja formalnih simbolov s pojasnili v angleščini, nemščini in slovenščini bosta dodana na koncu.

**This essay is a private author's work and no part of it may be used, reproduced or translated in any manner whatsoever without written permission except in the case of brief quotations embodided in critical articles and reviews.*

One can see in Heidegger's concern for humble things a continuation of his interest in the heritage of marginal practices. He now sees them as possibilities that have saving power precisely because they have never been taken seriously by the metaphysical tradition. Such practices, which have not been singled out as important and so technologized, provide a basis for resisting the technological understanding of being.

—Hubert L. Dreyfus and
Jane Rubin [BIW] 338-339

This essay is a preliminary study of the possibilities of a philosophical text formalization. It pertains to the phenomena of understanding within the framework of philosophy of Being or, precisely, within the Heidegger's Being-there [SZ, § 31]. As a first approximation of informational investigation, formulas corresponding to sentences of the Heideggerian text can be joined in a perplexed way, decomposed and composed, universalized and again particularized. In this mode the concept of Being-there as understanding can be depicted and developed formally obtaining some particular conceptualizations for possible later construction, design, and technology of an understanding system, that is, informational machine. So to say, we are stepping into the realm to make philosophical sentences informational in a symbolic (mathematical) way. One of the basic questions remains, does the possibility of an informational machine for philosophizing in accord to Being-there as understanding (and vice versa) already exist and what would the necessary approach to come closer to the realization of such concept be at all? The carefully reader will possibly find hints, kinks, and his own disclosedness of the problem of understanding within the philosophy of Being in general and within the Being-there in particular. Informational formalization of philosophy in question will certainly brighten the domain of understanding.

1. INTRODUCTION

To get a right approach to understanding it is essential at the outset not to think of understanding as a cognitive phenomenon. ... For Heidegger primordial understanding is know-how. ... understanding a hammer at its most primordial means knowing how *to hammer.*

—Hubert L. Dreyfus [BIW] 184

The phenomenality of understanding belongs to the main stream of the postmodernistic philosophic, scientific, and technological movement. Irrespective of the philosophic doubt to disclose ever the phenomenality of understanding and make it a scientific and technological tool, the study of intelligence in living organisms remains the most disturbing and irritating view of the future research. The phenomenality of understanding opens a sufficiently broad and unexplored realm of cognition, intelligence, reason, and mind.

The question we put into consideration is, how do philosophers comprehend the phenomena of understanding and which concepts (knowledge, beliefs, world views, rarely notions) do they use in the disputes concerning understanding, interpretation, explanation, and so forth. In this sense, our challenge is to examine some parts of the concerned philosophic disputes in an informational way. In experiments like these, we can study, luckily, some paragraphs of the text written by Martin Heidegger, who, as it seems, has determined understanding not only in the most lucid and complex way till now, but, maybe unconsciously, also in an appropriate informational way. Our attempt will be to prove this assertion in a consequent and formal way.

The basic question of the present investigation will be how does the system of understanding, which Heidegger harnessed together in the realm of the philosophy of Being and time, inform as an informational entity, that is, as a literary symbolic, operand-operator, open informational system. This investigation can offer several hints for the top-down (or, according to K. Popper, from the view of the third world) cognition and construction of comprehensive, intelligent, and understanding systems. To remind the reader, Popper [OK] says that the word *world* or *universe* must not be comprehended too seriously and, in this manner, three worlds or universes can be distinguished: (1) the world of physical objects or states; (2) the world of conscious or mental states, or maybe of ability to act; and (3) the world of objective contents of thought, especially scientific and poetic thought and art works (acts). In the third world there are,

for instance, theoretical systems, problems and problem situations, critical arguments, states of discourse and, certainly, the contents of newspapers, books, and libraries.

When I began to write my first essay on informational phenomenality, in spring 1987 [OWI], I was silently hopping that one day I will be able to put the Heideggerian Being-there (Dasein) as understanding [SZ, §31] into a form of the arising formal interpretation. Now, as this possibility dawned, my task is to prove the potentiality and appropriateness of the so-called formal informational language. A formalistic effort concerning understanding was already invested in the essay *Understanding as Information II* [UAI2], where some Heideggerian views of the loseableness and loosing of understanding have been shown by means of a formal system of informational formulas.

In this essay I will use the technique of formal informational interpretation of sentences belonging to the Heideggerian text. Each paragraph of the chosen text [BT, §31] will be interpreted by a kernel (in some way background) informational system of formulas as an informationally arising entity. Around such a system additional formulas will occur detailing (and perplexing) the interpretation of the system. If a sentence will not be taken into the interpretative consideration (parenthetical or informationally suppressed matter), it will be (temporarily) enclosed into brackets. The reader will come into the position to experience how formal informational interpretations can backwardly influence the human understanding of the original text, to enrich it informationally in a potentially understanding sense. This possibility will arise because of the general and imaginatively unbounded nature of informational operator $\models$ representing the entirety of an informational background.

At the end of the essay, the reader will find two dictionaries ordered by the informational symbols (occurring operands and operators), corresponding English terms, and their translation into German and Slovene. Thus, the opportunity will be given to make further comparisons of meaning and understanding of formulas interpreting the text in English, German, and Slovene.

2. A FORMAL INFORMATIONAL INTERPRETATION OF BEING--THERE AS UNDERSTANDING

Rather, modern natural science, modern mathematics, and modern metaphysics sprang from the same root of the mathematical in the wider sense. Because metaphysics, of these three, reaches farthest—to what is, in totality—and because at the same time it also reaches deepest toward the being of what is as such, therefore it is precisely metaphysics which must dig down to the bedrock of its mathematical base and ground.

—Martin Heidegger [WIT] 97-98

2.0. *THE SEGMENTATION AND STUDY OF THE CHOSEN TEXT*

In this chapter we deal with the verbal and symbolic interpretation of the text belonging to section §31 (Being-there as Understanding) in Heidegger's *Sein und Zeit* [SZ, 142-148; BT, 182-188; BV, 162-168]. The text of section §31 [BT] includes 18 paragraphs which will be analyzed and formally developed in a sentence by sentence fashion. We shall number the sentences of each paragraph by bracketed markers [paragraph_number.sentence_number] and to each sentence corresponding formula by parenthesized markers (paragraph_number.sentence_number). The Heideggerian terms of Being will be used consequently as particular informational entities (headwords, passwords); the reader can find an index of German and English expressions (headwords) including short notes in [BT]. On the basis of this index, the reader can study the concepts of particular terms and see how complex the most of them are conceptualized. Each term is an informational system of formulas and terms are mutually perplexed in various informational ways. By these initial comments, we can proceed by the procedure of study and text formalization in a subsequent and systematic way.

2.1. *THE FIRST PARAGRAPH OF §31 [BT]*

[1.1] State-of-mind is *one* of the existential structures in which the Being of the 'there' maintains itself.

The operands are: $\mathfrak{S}_{mind}$ marks state-of-mind; σ_{exist} marks existential structures; $\mathfrak{B}_{there}$ marks the Being-there (the Being of the 'there'). The formal interpretation of the sentence is

(1.1) $\mathfrak{S}_{mind} \in \sigma_{exist}$;
$(\mathfrak{B}_{there} \models \sigma_{exist}) \models \mathfrak{B}_{there}$

The explanation of operators: $\in$ marks the operation *is one of*. The part of the sentence (meaningly unmodified) ... *the Being of the 'there'* [$\mathfrak{B}_{there}$] *maintains itself* ... within the existential structures σ_{exist} is formalized in the second line of (1.1) by the so-called metaphysical (circular) form for operand $\mathfrak{B}_{there}$. □

[1.2] Equiprimordial with it in constituting this Being is *understanding*.

The exact meaning of this sentence is the following: Equiprimordial with state of mind in constituting the Being-there is *understanding*.

The new operand is $\mathfrak{U}$ which marks understanding. The formal interpretation of the sentence is

(1.2) $(\mathfrak{S}_{mind}, \mathfrak{U} \models_{const} \mathfrak{B}_{there}) \models_{equi\_p}$

This formula explains the fact that entities $\mathfrak{S}_{mind}$ and $\mathfrak{U}$ constitute (operator $\models_{const}$) entity $\mathfrak{B}_{there}$ in an openly equiprimordial way. To be equiprimordial (gleichurspünglich) means to inform in an equiprimordial way (operator $\models_{equi\_p}$). The comma between two informational operands has the meaning of 'the one and the other operand' (parallelism). □

[1.3] A state-of-mind always has its understanding, even if it merely keeps it suppressed.

The meaning of this sentence is, that a state-of-mind always contains its understanding (in this case $\mathfrak{U}$) and informs this fact in a suppressed manner.

The formal interpretation of the last sentence could be

(1.3) $(\mathfrak{U} \subset_{always} \mathfrak{S}_{mind}) \models_{supp}$

Operator $\subset_{always}$ is a particular time operator and means 'always has' or 'always contains'. The suppressed informing of this containment remains open. □

[1.4] Understanding always has its mood.

If $\mathfrak{M}$ marks the mood, the formula for this sentence is

(1.4) $\mathfrak{M}(\mathfrak{U}) \subset_{always} \mathfrak{U}$

No further comment on this formula is necessary. □

[1.5] If we Interpret understanding as a fundamental *existentiale*, this indicates that this phenomenon is conceived as a basic mode of Dasein's *Being*.

Let us expand the meaning of this sentence in the following way. The first part of the sentence says that we interpret (understand) understanding as a fundamental existentiale. And if so, then this indicates that this phenomenon (our understanding of understanding) is conceived (understood) as a basic mode of Dasein's Being. The sentence as a whole is implicative. Certainly, the we in this sentence can have the function of Dasein's Being and need not be treated as a separate operand. Evidently, Dasein's Being, marked by $\mathfrak{B}_{\mathfrak{D}}$ (or, also, Being of Dasein, $\mathfrak{B}(\mathfrak{D})$), is a component of Dasein $\mathfrak{D}$ and there exist the so-called basic modes μ_{basic} of Dasein's Being. Further, a fundamental existentiale ε_{fund} is the coming out or meaning produced by understanding. In whole, there is

(1.5) $\mathfrak{B}_{\mathfrak{D}} \subset \mathfrak{D}$; $\mu_{basic} \in \mathfrak{B}_{\mathfrak{D}}$;
$((\mathfrak{B}_{\mathfrak{D}} \models_{int} \mathfrak{U}) \models \varepsilon_{fund}) \Rightarrow (\mu_{basic} \in \mathfrak{B}_{\mathfrak{D}})$

Operator $\models_{int}$ has the meaning of interpretative informing and operator $\Rightarrow$ marks the informational implication. Formula (1.5) can be refined (informationally completed, supplemented) in several ways. □

[1.6] On the other hand, 'understanding' in the sense of *one* possible kind of cognizing among others (as distinguished, for instance, from 'explaining'), must, like explaining, be Interpreted as an existential derivative of that primary understanding which is one of the constituents of the Being of the "there" in general.

The meaning of the 'on the other hand' is 'in parallel'. Formulas, separated by a semicolon, are

understood always to exist in parallel. Understanding $\mathfrak{U}$ as an entity is only one possible kind of cognizing $\mathfrak{K}_{cogn}$ among other kinds of understanding or there exist always other kinds of understanding. But understanding is interpreted as an existential derivative δ_{exist} of the very primary understanding $\mathfrak{U}_{prim}$ which constitutes (roots in) (operator $\subset$ or $\subset_{const}$ can be chosen) the Being of "there". Thus, Being of "there" itself can be conceived as the producer of this sort of primary understanding. Furthermore, understanding $\mathfrak{U}$ is distinguished (operator $\neq$) from explaining $\mathfrak{E}_{expl}$. Within this view, we can put down the formal system

$$(1.6)\quad \begin{array}{l}((\mathfrak{U} \in \mathfrak{K}_{cogn}) \models_{int} \delta_{exist}) \subset \mathfrak{U}_{prim};\\ \mathfrak{B}_{there} \models (\mathfrak{U}_{prim} \subset \mathfrak{B}_{there});\\ \mathfrak{U} \neq \mathfrak{E}_{expl}\end{array}$$

The last system of formulas ends the formalization procedure of the sentences of the first paragraph in chapter §31 [BT]. It is to stress that the listed formulas phenomenalizing the sentences of the first paragraph are in no way complete informational systems and can be supplemented in a developmentally and explanatory open way, so at some later state, they can even informationally exceed the contents of the particular original sentences. □

2.2. *THE SECOND PARAGRAPH OF §31 [BT]*

[2.1] [We have, after all, already come up against this primordial understanding in our previous investigations, though we did not allow it to be included explicitly in the theme under discussion.]

This sentence is a kind of comment which concerns the previous text. □

[2.2] To say that in existing, Dasein is its "there", is equivalent to saying that the world is "there"; its *Being-there* is Being-in.

We take the following informational interpretation: in informing, Dasein $\mathfrak{D}$ is the there τ_{there} of Dasein $\mathfrak{D}$, marked by $\tau(\mathfrak{D}_{there})$; this is informationally equivalent to the formula that the world $\mathfrak{W}_{world}$ informs as the there τ_{there}. The world's Being-there $\mathfrak{B}_{there}(\mathfrak{W}_{world})$ informs as Being-in $\mathfrak{B}_{in}$. Informing of Dasein can be symbolized explicitly by the Dasein formula system $(\mathfrak{D} \models; \models \mathfrak{D})$. Thus,

$$(2.2)\quad \begin{array}{l}((\mathfrak{D} \models; \models \mathfrak{D}) \models \tau_{there}(\mathfrak{D}))\\ \Leftrightarrow (\mathfrak{W}_{world} \models \tau_{there});\\ \mathfrak{B}_{there}(\mathfrak{W}_{world}) \models \mathfrak{B}_{in}\end{array}$$

The last formula system is rather generalized than reduced in comparison to the second sentence of the second paragraph of §31 [BT]. □

[2.3] And the later is likewise 'there', as that for the sake of which Dasein is.

The later in the last sentence concerns the Being-in, that is, operand $\mathfrak{B}_{in}$. Further, the 'is likewise' corresponds to an informational operator and, if we take the most general case, $\models$, this operator can always be particularized in an adequate way. The sequence 'for the sake of which' points to a causal situation, that is, implication $\Rightarrow$, and by the 'Dasein is' the existential (in our language, informational) nature of Dasein is stressed, which can be symbolized explicitly by $(\mathfrak{D} \models; \models \mathfrak{D})$. The formula for this sentence becomes

$$(2.3)\quad (\mathfrak{B}_{in} \models \tau_{there}) \Rightarrow (\mathfrak{D} \models; \models \mathfrak{D})$$

This completes the symbolic interpretation of the sentence. □

[2.4] In the "for-the-sake-of-which", existing Being-in-the-world is disclosed as such, and this disclosedness, we have called "understanding".

This sentence represents a definition of understanding which is the disclosedness concerning formula (2.3) as well as formula (2.2). These formulas build up the informational cycle of Dasein in which the there, world, Being-there, and Being-in are constitutive components. The disclosedness of understanding is a kind of informing of Being-in-the-world within this cycle. Let us mark the "for-the-sake-of-which" by φ_{sake}. Thus, the last sentence induces the formal system as a consequence of the previous two sentences, that is,

$$(2.4)\quad \begin{array}{l}((\mathfrak{B}_{in\text{-}the\text{-}world} \models_{discl} \mathfrak{U}) \subset \varphi_{sake};\\ \varphi_{sake} \Rightarrow\\ ((((\mathfrak{D} \models; \models \mathfrak{D}) \models \tau_{there}(\mathfrak{D})) \Leftrightarrow\\ \quad (\mathfrak{W}_{world} \models \tau_{there}));\\ \quad (\mathfrak{B}_{there}(\mathfrak{W}_{world}) \models \mathfrak{B}_{in});\end{array}$$

$$((\mathfrak{B}_{in} \models \tau_{there}) \Rightarrow (\mathfrak{D} \models; \models \mathfrak{D})))$$

To say that something is disclosed means that something informs (the disclosedness) (operator $\models_{discl}$). A further interweavement of formulas simulating particular sentences will be discussed in a later section. □

[2.5] In the understanding of the "for-the-sake-of-which", the significance which is grounded therein, is disclosed along with it.

This operand marks a joined system of formulas (2.2) and (2.3). Understanding $\mathfrak{U}$ understands φ_{sake}, that is, as $\mathfrak{U}(\varphi_{sake})$. Within this understanding, understanding $\mathfrak{U}$ is informed by the "for-the-sake-of-which" φ_{sake}, that is, $\varphi_{sake} \models \mathfrak{U}$. The significance of understanding is marked by a generalized operand ξ_{sign}. Under these circumstances, there is,

(2.5) $\quad (\xi_{sign} \subset \mathfrak{U}(\varphi_{sake})) \models_{discl}$

This formula includes formulas (2.2) and (2.3) through the operand, marked by φ_{sake}. □

[2.6] The disclosedness of understanding [$\mathfrak{U} \models_{discl}$], as [$\models_{as}$] the disclosedness of the "for-the-sake-of-which" [$\mathfrak{U}(\varphi_{sake}) \models_{discl}$] and of significance [$\xi_{sign}$] equiprimordially, pertains [$\rightarrow_{equi\_p}$] to the entirety of Being-in-the-world [$\mathfrak{B}_{in\text{-}the\text{-}world}$].

In the last sentence we introduced the bracketed expressions right within the text of the sentence to achieve a direct correspondence between the word notions and symbols. Thus, we put immediately,

(2.6) $\quad ((\mathfrak{U} \models_{discl}) \models_{as}$
$(\mathfrak{U}(\varphi_{sake} \models_{discl}; \xi_{sign} \models_{discl}))$
$\rightarrow_{equi\_p} \mathfrak{B}_{in\text{-}the\text{-}world}$ □

[2.7] Significance [ξ_{sign}] is that on the basis of which the world [$\mathfrak{W}_{world}$] is disclosed as such.

This sentence yields

(2.7) $\quad \mathfrak{W}_{world}(\xi_{sign}) \models_{discl} \mathfrak{W}_{world}$ □

[2.8] To say that the "for-the-sake-of-which" [φ_{sake}] *and* significance [ξ_{sign}] are both disclosed in Dasein [$\varphi_{sake}, \xi_{sign} \subset_{discl} \mathfrak{D}$], means [$\Leftrightarrow$] that Dasein [$\mathfrak{D}$] is that entity [$\mathfrak{D} \models; \models \mathfrak{D}$] which, as Being-in-the-world [$\mathfrak{B}_{in\text{-}the\text{-}world}$], is an issue for itself [$\mathfrak{D} \models \mathfrak{D}$].

One can put

(2.8) $\quad (\varphi_{sake}, \xi_{sign} \subset_{discl} \mathfrak{D}) \Leftrightarrow$
$((\mathfrak{D} \models_{as} \mathfrak{B}_{in\text{-}the\text{-}world}) \models \mathfrak{D})$

This formula can be additionally interpreted by the corresponding German sentence: Worumwillen *und* Bedeutdsamkeit sind im Dasein erschlossen, besagt: Dasein is Seiendes [$\mathfrak{D} \models; \models \mathfrak{D}$], dem es als In-der-Welt-sein um es selbst geht. □

2.3. *THE THIRD PARAGRAPH OF § 31 [BT]*

[3.1] When we are talking ontically we sometimes use [$\models_{ont,some}$] the expression 'understanding something' [$\mathfrak{U}(\alpha)$] with the significance [meaning μ_{sign}] of 'being able to manage something' [$\models_{able\_man}\ \alpha$], 'being a match for it' [$\models_{match}\ \alpha$], 'being competent to do something' [$\models_{comp}\ \alpha$].

Formula for this sentence is

(3.1) $\quad \models_{ont,some}(\mathfrak{U}(\alpha) \models (\mu_{sign}(\models_{able\_man}\ \alpha),$
$\mu_{sign}(\models match\ \alpha), \mu_{sign}(\models_{comp}\ \alpha)))$

This formula is open as the 'we' or an informational entity (operand) is not considered explicitly on the left side of operator $\models_{ont,some}$. □

[3.2] In [$\subset$] understanding [$\mathfrak{U}$], as an *existentiale* [ε_{exist}], that which we have such competence over [something, that is, $\models_{comp}\ \alpha$ or, in the form of a process, $\mathfrak{G}_{known}$] is not [$\not\models$] a "what" [$\mathfrak{W}_{what}$], but [symbol ';' or, more definite, operator $\models_{but}$] Being [$\mathfrak{B}$] as existing [that is, $\mathfrak{B} \models; \models \mathfrak{B}$ or, explicitly, $\mathfrak{E}_{exist}$].

Evidently, the last part of the sentence is an informational inclusion in regard to the first part. So,

(3.2) $\quad (\mathfrak{W}_{what} \not\models_{comp} \alpha; \mathfrak{B} \models_{as} (\mathfrak{B} \models; \models \mathfrak{B})) \subset$
$(\mathfrak{U} \models_{as} \varepsilon_{exist})$

Another interpretation of this sentence could be, for instance,

(3.2') $\quad (((\varepsilon_{exist} \models \mathfrak{G}_{known}) \subset \mathfrak{U}) \not\models \mathfrak{W}_{what})$
$\models_{but} (\mathfrak{B} \models_{as} \mathfrak{E}_{exist})$

To these formulas a different formula can be constructed from the sentence in German: Das im Verstehen als Existenzial Gekonnte [$\mathfrak{G}_{known}$] ist kein Was, sondern [$\models_{but}$] das Sein als Existieren. For this sentence the following formula could be, in principle, adequate:

(3.2") $((\varepsilon_{exist} \subset (\mathfrak{U} \models_{as} \mathfrak{G}_{known})) \not\models \mathfrak{W}_{what})$
$\models_{but} (\mathfrak{B} \models_{as} (\mathfrak{B} \models; \models \mathfrak{B}))$

The most straightforward formalizing approach of the German sentence would be

(3.2^3) $(\mathfrak{G}_{known} \models_{as} \varepsilon_{exist}) \subset \mathfrak{U}$;
$\mathfrak{G}_{known} \not\models \mathfrak{W}_{what}$;
$\mathfrak{G}_{known} \models (\mathfrak{B} \models_{as} \mathfrak{E}_{exist})$

These examples show the difficulties which may occur at the translation of sentences from one language into another or, speaking informationally, differences which may originate at or depend on the place of observation. □

[3.3] The kind [$\mathfrak{K}$] of Being [$\mathfrak{B}$] which Dasein [$\mathfrak{D}$] has [$\subset$], as potentiality-for-Being [$\models_{as} \pi_{for\text{-}Being}$], lies existentially in [$\subset_{exist}$] understanding [$\mathfrak{U}$].

Formula of this sentence is

(3.3) $((\mathfrak{K}(\mathfrak{B}) \subset \mathfrak{D}) \models_{as} \pi_{for\text{-}Being}) \subset_{exist} \mathfrak{U}$

The German sentence is: Im Verstehen liegt existenzial die Seinsart des Daseins als Sein-können. This sentence delivers, for instance, the formula

(3.3') $(\mathfrak{K}_{Being}(\mathfrak{D}) \models_{as} \pi_{for\text{-}Being}) \subset_{exist} \mathfrak{U}$

where 'die Seinsart des Daseins' is marked by $\mathfrak{K}_{Being}(\mathfrak{D})$. As we see, the English translation 'the kind of being which Dasein has' gives a different symbolic expression. □

[3.4] Dasein [$\mathfrak{D}$] is not something present-at-hand [$\not\models \alpha_{present\text{-}at\text{-}hand}$] which possesses its competence [$\subset_{comp}$] for something [α] by way of an extra [$\models_{extra}$]; it [$\mathfrak{D}$] is primarily [$\models_{prim}$] Being-possible [$\mathfrak{B}_{possible}$].

This sentence is a system of two formulas:

(3.4) $(\alpha \subset_{comp} (\mathfrak{D} \not\models \alpha_{present\text{-}at\text{-}hand})) \models_{extra}$;
$(\mathfrak{D} \models_{prim} \mathfrak{B}_{possible})$ □

[3.5] Dasein [$\mathfrak{D}$] is in every case what it can be [$\mathfrak{D} \models; \models \mathfrak{D}$], and in the way in which it is [$\models$] its possibility [π].

When we say that something is in every case what it can be, we use a kind of determination, for instance, the defining equivalence [$\Leftrightarrow_{Df}$] or simply the sign of informing [$\models$]. So, let it be

(3.5) $(\mathfrak{D} \models (\mathfrak{D} \models; \models \mathfrak{D})) \models \pi(\mathfrak{D})$

or

(3.5') $(\mathfrak{D} \Leftrightarrow_{Df} (\mathfrak{D} \models; \models \mathfrak{D})) \models \pi(\mathfrak{D})$ □

[3.6] The Being-possible [$\mathfrak{B}_{possible}$] which is essential for [$\models_{essen\_for}$] Dasein [$\mathfrak{D}$], pertains [$\models_{pertains}$] to the ways of its solicitude [$\pi_{solicitude}(\mathfrak{D})$] for Others [$\omega$] and of its [$\mathfrak{D}$] concern [$\models_{concern}$] with the 'world' [$\mathfrak{W}_{world}$], as we have characterized [$\dashv\models_{char}$] them; and in all these, and always [$\subset_{always}$], it pertains to Dasein's potentiality-for-Being [$\pi_{for\text{-}Being}(\mathfrak{D})$] towards itself, for the sake of itself.

Let us write down the formal approximation of the last sentence as the following:

(3.6) $((\mathfrak{D} \models_{pertain} \pi_{for\text{-}Being}) \models_{pertain} \mathfrak{D}) \subset_{always}$
$((\mathfrak{B}_{possible} \models_{essen\_for} \mathfrak{D}) \models_{pertain}$
$(((\pi_{solicitude}(\mathfrak{D}) \models_{for} \omega);$
$(\pi(\mathfrak{D}) \models_{concern} \mathfrak{W}_{world})) \dashv\models_{char}))$

The corresponding German sentence cän deliver another illumination of the sentence understanding. It is: Das wesenhafte Möglich-sein des Daseins betrifft die charakterisierten Weisen des Besorgens der »Welt«, der Fürsorge für die anderen und im all dem und immer schon das Seinkönnen zu ihm selbst, umwillen seiner. Here, the explaining from the end of the sentence confirms to major extent the sense of formula (3.6).

The problem of translation from one to another language (English, German, informational) becomes also evident on the formal (or informational) level. □

[3.7] The Being-possible [$\mathfrak{B}_{possible}$] which Dasein [$\mathfrak{D}$] is existentially in every case [$\models_{exist}$], is to be sharply distinguished [$\neq_{sharply}$] both from empty

logical possibility [$\pi_{log} \models_{empty}$] and from the contingency [γ_{cont}] of something present-at-hand [$\alpha_{present\text{-}at\text{-}hand}$], so far as [$\Leftarrow_{so\text{-}far\text{-}as}$] with the present-at-hand this or that can 'come to pass' [$\alpha \models \alpha_{present\text{-}at\text{-}hand}$].

Accordingly to this sentence one can put

(3.7) $(((\mathfrak{D} \subset \mathfrak{B}_{possible}) \models_{exist}) \neq_{sharply} ((\pi_{log} \models_{empty}), \gamma_{cont}(\alpha_{present\text{-}at\text{-}hand}))) \Leftarrow_{so\text{-}far\text{-}as} (\alpha \models \alpha_{present\text{-}at\text{-}hand})$

This formula is an informational implication of the type 'so-far-as'. Considering the original German sentence which in its first part says Das Möglichsein, das je das Dasein existenzial ist, ..., could deliver ($\mathfrak{B}_{possible} \models_{exist} \mathfrak{D}$) instead of (($\mathfrak{D} \subset \mathfrak{B}_{possible}$) $\models_{exist}$) in formula (3.7). □

[3.8] As [$=\!|_{as}$] a modal category [γ_{modal}] of presence-at-hand [$\pi_{at\text{-}hand}$], possibility [π] signifies [$\models_{sign}$] what [α] is *not yet* actual [$\not\models_{act}$] and what is *not at any time* necessary [$\not\models \nu_{at\_any\_time}$].

The informational formula for this sentence can be the following:

(3.8) $(\gamma_{modal}(\pi_{at\text{-}hand})) =\!|_{as} (\pi \models_{sign} ((\alpha \not\models_{act}), (\alpha \not\models \nu_{at\_any\_time}))$ □

[3.9] It [π] characterizes [$\models_{char}$] the *merely* possible [α_{merely_poss}].

In short, there is,

(3.9) $\pi \models_{char} \alpha_{merely\_poss}$ □

[3.10] Ontologically [$\models_{ont}$] it [π] is on a lower level [λ_{lower}] than [$\models_{than}$] actuality [α_{act}] and necessity [ν].

A direct translation of this sentence into informational language is

(3.10) $(\pi \models_{ont} \lambda_{lower}) \models_{than} (\alpha_{act}, \nu)$ □

[3.11] On the other hand [$\|\models$], possibility [π] as an *existentiale* [ε_{exist}] is the most primordial and ultimate positive way [υ_{way}] in which Dasein [$\mathfrak{D}$] is characterized [$\models_{char}$] ontologically [$\models_{ont}$].

For this sentence one can set

(3.11) $(\pi \models_{as} \varepsilon) \| \models \upsilon_{way}$;

$((\models_{char} \mathfrak{D}) \subset \upsilon_{way}) \models_{ont}$ □

[3.12] As with existentiality in general [$\mathfrak{S}_{exist} \models$; $\models \mathfrak{S}_{exist}$], we can, in the first instance, only prepare for the problem of possibility [π].

The simplest way to interpret this sentence is formula

(3.12) $\pi \models_{as} (\mathfrak{S}_{exist} \models; \models \mathfrak{S}_{exist})$

In German, this sentence is: ... zunächst kann sie [π] wie Existenzialität überhaupt lediglich als Problem vorbereitet werden. The existentiality as a problem of possibility can be understood within the informing of the system $\mathfrak{S}_{exist} \models; \models \mathfrak{S}_{exist}$. □

[3.13] The phenomenal basis [φ_{basis}] for seeing it [π] at all is provided [$\models_{prov}$] by the understanding [$\mathfrak{U}$] as a disclosive potentiality-for-Being [$\pi_{for\text{-}Being}$].

The adequate, universalized formula to this sentence is, for instance,

(3.13) $(\mathfrak{U} \models_{as} \pi_{for\text{-}Being}) \models_{prov} ((\varphi_{basis} \models \pi) \models \varphi_{basis})$

The phrase 'for seeing it at all' remains hidden in the process $(\varphi_{basis} \models \pi) \models \varphi_{basis}$. □

2.4. *THE FOURTH PARAGRAPH OF § 31 [BT]*

[4.1] Possibility [π], as an *existentiale* [ε_{exist}], does not signify [$\not\models_{sign}$] a free-floating potentiality-for- Being [$\pi_{for\text{-}Being} \models; \models \pi_{for\text{-}Being}$] in the sense [$\sigma_{sense}$] of the 'liberty of indifference' [$\lambda_{liber}(\iota_{indiff})$] (*libertas indifferentiae*).

The formula is

(4.1) $(\pi \models_{as} \varepsilon_{exist}) \not\models_{sign} ((\pi_{for\text{-}Being} \models; \models \pi_{for\text{-}Being}) \models \sigma_{sense}(\lambda_{liber}(\iota_{indiff})))$ □

[4.2] In every case Dasein [$\mathfrak{D}$], as essentially having [$\subset_{essen}$] a state-of-mind [$\mathfrak{S}_{mind}$], has already got [$\models_{already}$] itself into definite possibilities [π_{def}].

The rough formula for this sentence is

(4.2) $(\mathfrak{S}_{mind} \subset_{essen} \mathfrak{D}) \models_{already} (\mathfrak{D} \subset \pi_{def})$

where we did not consider the entity 'in every case' which could be expressed, for instance, by operator $\forall$ at the end of formula (4.2). It is to mention that the German sentence: Das Dasein ist als wesenhaft befindliches ... gives also a 'direct' formula $\mathfrak{D} \models_{as\_essen} \mathfrak{S}_{mind}$. □

[4.3] As the potentiality-for-Being [$\pi_{for\text{-}Being}$] which is *is* [$\pi_{for\text{-}Being} \models; \models \pi_{for\text{-}Being}$], it [$\mathfrak{D}$] has let such possibilities [$\pi(\pi_{for\text{-}Being})$] pass by [$\models_{pass\_by}$]; it is constantly waiving [$\models_{waive}$] the possibilities of its Being [$\pi(\mathfrak{B}(\mathfrak{D}))$], or else [in parallel] it seizes [$\models_{seize}$] upon them [$\pi(\mathfrak{B}(\mathfrak{D}))$] and makes [$\models$] mistakes [$\mu_{mistake}$].

For this sentence we get a system of formulas, that is,

(4.3) $(\mathfrak{D} \models_{as} (\pi_{for\text{-}Being} \models; \models \pi_{for\text{-}Being})) \models_{pass\_by} ((\pi_{for\text{-}Being}); \mathfrak{D} \models_{waive} \pi(\mathfrak{B}(\mathfrak{D})); (\mathfrak{D} \models_{seize} \pi(\mathfrak{B}(\mathfrak{D}))) \models \mu_{mistake}$ □

[4.4] But this [formula (4.3) marked by $\varphi_{(4.3)}$] means [$\Leftrightarrow$] that Dasein is Being-possible [$\mathfrak{D} \models \mathfrak{B}_{possible}$] which has been delivered [$\models_{deliver}$] over to itself [$\mathfrak{D}$]—*thrown possibility* [π_{thrown}] through and through [$\models_{through}$].

Thus, the resulting formula is

(4.4) $\varphi_{(4.3)} \Leftrightarrow ((((\mathfrak{D} \models \mathfrak{B}_{possible}) \models_{deliver} \mathfrak{D}) \models \pi_{thrown})) \models_{through}$

This formula is nothing else than an explanation of formula (4.3), marked by $\varphi_{(4.3)}$. The original German sentence, for comparison, is: Das besagt aber: das Dasein ist ihm selbst überantwortetes Möglichsein, durch und durch *geworfene Möglichkeit.* □

[4.5] Dasein [$\mathfrak{D}$] is the possibility [π] of Being-free [$\mathfrak{B}_{free}$] *for* its ownmost potentiality-for-Being [$\pi_{for\text{-}Being}(\mathfrak{D})$].

The formula for the last sentence is

(4.5) $(\mathfrak{D} \models \pi(\mathfrak{B}_{free})) \models_{for} \pi_{for\text{-}Being}(\mathfrak{D})$ □

[4.6] Its [$\mathfrak{D}$] Being-possible [$\mathfrak{B}_{possible}(\mathfrak{D})$] is transparent [$\models_{trans}$] to itself [$\mathfrak{D}$] in different possible ways and degrees [α_{poss_ways}].

The formula for this sentence is

(4.6) $(\mathfrak{B}_{possible}(\mathfrak{D}) \models_{trans} \mathfrak{D}) \models \alpha_{poss\_ways}$ □

2.5. *THE FIFTH PARAGRAPH OF § 31 [BT]*

[5.1] Understanding [$\mathfrak{U}$] is the Being [$\mathfrak{B}$] of such potentiality-for-Being [$\pi_{for\text{-}Being}$], which is never [$\not\models$] something still outstanding [α_{still_out}] as not yet [$\models_{as\_not\_yet}$] present-at-hand [π_{at_hand}], but which [in parallel] as something [α] which is essentially never [$\not\models_{essen}$] present-at-hand, *'is'* with [$\models_{with}$] the Being of Dasein [$\mathfrak{B}(\mathfrak{D})$], in the sense of existence [σ_{exist}].

The frame system of formulas for this sentence is

(5.1) $(\mathfrak{U} \models \mathfrak{B}(\pi_{for\text{-}Being})) \not\models (\alpha_{still\text{-}out} \models_{as\_not\_yet} \pi_{at\_hand}); ((\mathfrak{B}(\pi_{for\text{-}Being}) \models_{as} \alpha) \not\models_{essen} \pi_{at\_hand}) \models_{with} (\mathfrak{B}(\mathfrak{D}) \subset \sigma_{exist})$ □

[5.2] Dasein is such that in every case [$\forall$] it has understood [$\models_{\mathfrak{U}}$] (or alternatively, [in parallel] not understood [$\not\models_{\mathfrak{U}}$]) that it is to be thus or thus [$\mathfrak{D} \models; \models \mathfrak{D}$].

Let the first approximation of this sentence be

(5.2) $(\mathfrak{D} \forall (\mathfrak{D} \models_{\mathfrak{U}}; \mathfrak{D} \not\models_{\mathfrak{U}})) \models (\mathfrak{D} \models; \models \mathfrak{D})$

Several objections can be made to the last formula. First operator $\forall$ could be particularized into $\models_{always}$ or even into $\models_{such\_t\_e\_c}$ with the meaning 'is_such_that_in_every_case'. Further, *to be thus or thus* pertaining to Dasein $\mathfrak{D}$ and understanding $\mathfrak{U}$ could be formally interpreted as

(5.2') $(\mathfrak{D} \forall (\mathfrak{D} \models_{\mathfrak{U}}; \mathfrak{D} \not\models_{\mathfrak{U}})) \models (((\mathfrak{D} \models \mathfrak{U}) \models \mathfrak{D}); (\mathfrak{D} \dashv (\mathfrak{U} \dashv \mathfrak{D})); ((\mathfrak{D} \not\models \mathfrak{U}) \not\models \mathfrak{D}); (\mathfrak{D} \not\dashv (\mathfrak{U} \not\dashv \mathfrak{D})))$

Certainly, other interpretations of the sentence are possible. □

[5.3] As such understanding [$\mathfrak{U}$] it [$\mathfrak{D}$] 'knows' [$\models_{know}$] *what* it [$\mathfrak{D}$] is capable [$\models_{cap}$]—that is,

what its potentiality-for-Being [$\pi_{\text{for-Being}}(\mathfrak{D})$] is capable of.

One of possible formal interpretations of this sentence is the following:

(5.3) $(\mathfrak{D} \models_{\text{as}} \mathfrak{U}) \models_{\text{know}} ((\mathfrak{D} \models_{\text{cap}}), \pi_{\text{for-Being}}(\mathfrak{D}) \models_{\text{cap}})$

As we see, the 'what' [$\mathfrak{W}_{\text{what}}$] in the capability of informing of $\mathfrak{D}$ is ignored, however, it can be explicitly considered within the structure of the previous formula, closing it to some extent in the following way:

(5.3') $(\mathfrak{D} \models_{\text{as}} \mathfrak{U}) \models_{\text{know}} ((\mathfrak{D} \models_{\text{cap}} \mathfrak{W}_{\text{what}}), \pi_{\text{for-Being}}(\mathfrak{D}) \models_{\text{cap}} \mathfrak{W}_{\text{what}})$ □

[5.4] This 'knowing' [$\mathfrak{R}_{\text{know}}(\mathfrak{D}) = \varphi_{(5.3)}$] does not first arise [$\not\models_{\text{first}}$] from an immanent self-perception [$\mathfrak{P}_{\text{imm}}(\mathfrak{D})$], but [in parallel] belongs [$\subset$] to the Being of the "there" [$\mathfrak{B}(\tau_{\text{there}})$], which is essentially [$\models_{\text{essential}}$] understanding [$\mathfrak{U}$].

This sentence sounds understandingly and can be immediately formalized as

(5.4) $\mathfrak{R}_{\text{know}}(\mathfrak{D}) \not\models_{\text{first}} \mathfrak{P}_{\text{imm}}(\mathfrak{D})$;
$\mathfrak{R}_{\text{know}}(\mathfrak{D}) \subset (\mathfrak{B}(\tau_{\text{there}}) \models_{\text{essential}} \mathfrak{U})$

Operand $\mathfrak{R}_{\text{know}}(\mathfrak{D})$ is a occurrence of formula $\varphi_{(5.3)}$ in which $\mathfrak{D}$ as understanding informs capably. □

[5.5] And only *because* Dasein, in understanding [$(\mathfrak{D} \models \mathfrak{U}) \models \mathfrak{D}$], is its "there" [$\tau_{\text{there}}(\mathfrak{D})$], *can* it go astray [$\models_{\text{astray}}$] and fail to recognize [$\models_{\text{fail\_to\_recognize}}$] itself.

The symbolic interpretation of this sentence is

(5.5) $(((\mathfrak{D} \models \mathfrak{U}) \models \mathfrak{D}) \models \tau_{\text{there}}(\mathfrak{D})) \Rightarrow$
$(\mathfrak{D} \models_{\text{astray}}; \mathfrak{D} \models_{\text{fail\_to\_recognize}} \mathfrak{D})$ □

[5.6] And in so far as understanding is *accompanied by* [$\models_{\text{accomp\_by}}$] state-of-mind [$\mathfrak{S}_{\text{mind}}$] and as such is existentially surrendered [$\models_{\text{exist\_surr}}$] to thrownness [$\tau_{\text{thrown}}$], Dasein has in every case already [$\forall_{\text{already}}$] gone astray [$\models_{\text{astray}}$] and failed to recognize [$\models_{\text{fail\_to\_recognize}}$] itself.

One of the adequate formulas to this sentence could be

(5.6) $((\mathfrak{U} \models_{\text{accomp\_by}} \mathfrak{S}_{\text{mind}}) \models_{\text{exist\_surr}} \tau_{\text{thrown}}) \Rightarrow$
$(\mathfrak{D} \forall_{\text{already}} (\mathfrak{D} \models_{\text{astray}} \mathfrak{D};$
$\mathfrak{D} \models_{\text{fail\_to\_recognize}} \mathfrak{D}))$ □

[5.7] In its [$\mathfrak{D}$] potentiality-for-Being [$\pi_{\text{for-Being}}$] it is therefore delivered [$\models_{\text{deliver}}$] over to the possibility of first finding [$\models_{\text{first\_find}}$] itself again in its possibilities [$\pi(\mathfrak{D})$].

Thus, the last sentence of the 5th paragraph becomes formally

(5.7) $(\mathfrak{D} \subset \pi_{\text{for-Being}}(\mathfrak{D})) \models_{\text{deliver}}$
$(\pi(\mathfrak{D} \models_{\text{first\_find}} \mathfrak{D}) \subset \pi(\mathfrak{D}))$

Thus, we have, in a framing manner, formalized the sentences of the fifth paragraph. □

2.6. *THE SIXTH PARAGRAPH OF § 31 [BT]*

[6.1] *Understanding* [$\mathfrak{U}$] *is* [$\models$] *the existential Being* [$\mathfrak{B}_{\text{exist}}$] *of Dasein's own potentiality-for-Being* [$(\mathfrak{D} \models \pi_{\text{for-Being}}(\mathfrak{D})) \models \mathfrak{D}$]*; and it* [$\mathfrak{U}$] *is so in such a way that this Being discloses* [$\models_{\text{discl}}$] *in itself* [$\mathfrak{B}_{\text{exist}}$] *what* [$\mathfrak{W}_{\text{what}}$] *its Being is capable of* [$\models_{\text{cap}}$].

This sentence leaves open various possibilities. One of them, as an initial situation, is, for instance,

(6.1) $((\mathfrak{U} \models \mathfrak{B}_{\text{exist}}((\mathfrak{D} \models \pi_{\text{for-Being}}(\mathfrak{D})) \models \mathfrak{D}))$
$\models_{\text{discl}} \mathfrak{B}_{\text{exist}}) \models_{\text{cap}} \mathfrak{W}_{\text{what}}(\mathfrak{B}_{\text{exist}})$

It is not for the first time that we use the operand form $\xi(\eta)$ instead of the explicit operator form $\xi \models_{\text{of}} \eta$. This convention comes probably from the mathematical way of thinking. However, we must be aware that in $\xi(\eta)$, entity η can take the form of an informational formula, arbitrarily complex, open, circular, etc. And this happens in case of $\mathfrak{B}_{\text{exist}}((\mathfrak{D} \models \pi_{\text{for-Being}}(\mathfrak{D})) \models \mathfrak{D}))$, where $\mathfrak{B}_{\text{exist}}$ is in position of ξ, etc. □

[6.2] [We must grasp the structure of this *existentiale* more precisely.]

This sentence is a comment on that what has to follow in the next paragraphs. □

2.7. *THE SEVENTH PARAGRAPH OF § 31 [BT]*

[7.1] As [$\models_{as}$] a disclosure [$\mathfrak{D}_{discl}$], understanding [$\mathfrak{U}$] always pertains to [$\forall_{pertain}$] the whole basic state [$\mathfrak{W}_{basic\_state}$] of Being-in-the-world [$\mathfrak{B}_{in\text{-}the\text{-}world}$].

A formula for this sentence is

(7.1) $(\mathfrak{U} \models_{as} \mathfrak{D}_{discl})\ \forall_{pertain}$
$\mathfrak{W}_{basic\_state}(\mathfrak{B}_{in\text{-}the\ world})$ □

[7.2] As [$\models_{as}$] a potentiality-for-Being [$\pi_{for\text{-}Being}$], any Being-in [$\mathfrak{B}_{in}$] is a potentiality-for-Being-in-the-world [$\pi_{for}(\mathfrak{B}_{in\text{-}the\text{-}world})$].

(7.2) $(\mathfrak{B}_{in} \models_{as} \pi_{for}(\mathfrak{B}_{in\text{-}the\text{-}world})$ □

[7.3] Not only is the world [$\mathfrak{W}_{world}$], *qua* world, disclosed [$\models_{discl}$] as possible significance [ξ_{sign_poss}], but [$\models_{but}$ or in parallel] when that [α] which is within-the-world [$\alpha \subset \mathfrak{W}_{world}$] is [$\Rightarrow$] itself freed [$\alpha \models_{free} \alpha$], this entity is freed for *its own* possibilities [$\pi(\alpha)$].

In this sentence two implicative operators appear, where the first one has the meaning of 'when ..., then ...' and the second one of 'if ..., then ...'. So,

(7.3) $((\mathfrak{W}_{world} \models_{as} \mathfrak{W}_{world}) \models_{discl} \xi_{sign\_poss})$
$\models_{but}$
$(((\alpha \subset \mathfrak{W}_{world}) \Rightarrow (\alpha \models_{free} \alpha)) \Rightarrow$
$(\pi(\alpha) \models_{free} \alpha))$ □

[7.4] That [α] which is ready-to-hand [$\mathfrak{R}_{to\text{-}hand}$] is discovered as [$\models_{discover\_as}$] such in its service*ability* [$\mathfrak{A}_{service}$], its us*ability* [$\mathfrak{A}_{use}$], and its detriment*ality* [$\mathfrak{A}_{detriment}$].

The appropriate formula is, for instance,

(7.4) $(\alpha \models_{as} \mathfrak{A}_{service}(\alpha);\ \alpha \models_{as} \mathfrak{A}_{use}(\alpha);$
$\alpha \models_{as} \mathfrak{A}_{detriment}(\alpha)) \models_{discover\_as}$
$(\alpha \models \mathfrak{R}_{to\text{-}hand})$ □

[7.5] The totality of involvements [$\tau_{total}(\iota_{involve})$] is revealed as [$\models_{reveal\_as}$] the categorial whole [$\mathfrak{W}_{categorial}$] of a *possible* interconnection [$\pi_{interconn}$] of the ready-to-hand [$\mathfrak{R}_{to\text{-}hand}$].

The corresponding formula is, for example,

(7.5) $(\mathfrak{W}_{categorial} \models_{reveal\_as} \tau_{total}(\iota_{involve}))$
$\models_{of} \pi_{interconn}(\mathfrak{R}_{to\text{-}hand})$ □

[7.6] But even the 'unity' [υ_{unity}] of the manifold [$\mu_{manifold}$] present-at-hand [$\pi_{at\text{-}hand}$], of Nature [$\mathfrak{N}$], can be discovered [$\models_{discover}$] only if a *possibility* of it [$\pi(\upsilon_{unity})$] has been disclosed [$\models_{discl}$].

An open formula for this sentence is

(7.6) $(\models_{discl} \pi(\upsilon_{unity}))) \Rightarrow$
$(\models_{discover} (\upsilon_{unity}(\mu_{manifold}(\pi_{at\text{-}hand})),$
$\upsilon_{unity}(\mathfrak{N})))$ □

[7.7] Is it [$\models_{quest}$] accidental [$\models_{accid}$] that the question [$\mathfrak{Q}_{question}$] about [$\models_{about}$] the *Being* of Nature [$\mathfrak{B}(\mathfrak{N})$] aims at [$\models_{aim\_at}$] the 'conditions of its *possibility*' [$\gamma_{cond}(\pi(\mathfrak{B}(\mathfrak{N})))$]?

This question informs accidentally, that is,

(7.7) $((\mathfrak{Q}_{question} \models_{about} (\mathfrak{B}(\mathfrak{N}) \models_{aim\_at}$
$\gamma_{cond}(\pi(\mathfrak{B}(\mathfrak{N}))))) \models_{accid}) \models_{quest}$ □

[7.8] On what [$\mathfrak{W}_{what}$] is such an inquiry [$\varphi_{(7.7)}$] based [$\models_{quest}$]?

The formal information of this sentence is

(7.8) $\varphi_{(7.7)} \models_{quest} \mathfrak{W}_{what}$ □

[7.9] When confronted [$\models_{confront}$] with this inquiry [$\varphi_{(7.7)}$], we can not leave aside the question: *why* are [$\models_{why}$] entities [α] which are not of the character of Dasein [$\gamma_{char}(\mathfrak{D})$] understood [$\models_{\mathfrak{U}}$] in their Being [$\mathfrak{B}(\alpha)$], if [$\Leftarrow$] they are disclosed [$\models_{discl}$] in accordance with the conditions [γ_{cond}] of their possibility [$\pi(\alpha)$]?

The formula for this sentence can become rather complicated, however, one of its formal approximations could be

(7.9) $(\models_{confront} (\varphi_{(7.7)} \models_{quest} \alpha)) \Rightarrow$
$((((\alpha \not\models \gamma_{char}(\mathfrak{D})) \models_{\mathfrak{U}} \mathfrak{B}(\alpha)) \Leftarrow$
$(\gamma_{cond}(\pi(\alpha)) \models_{discl} \alpha)) \models_{why})$ □

[7.10] [Kant presupposes something of the sort, perhaps rightly.]

This sentence is taken as an insignificant comment. □

[7.11] [But this presupposition itself is something that cannot be left without demonstrating how it is justified.]

In the next paragraphs this presupposition will be demonstrated. □

2.8. *THE EIGHTH PARAGRAPH OF § 31 [BT]*

[8.1] Why [$\mathfrak{W}_{why}$] does [$\models_{quest}$] the understanding [$\mathfrak{U}$]—whatever may be the essential dimension [ε_{dim}] of that which can be disclosed in [$\models_{discl}$] it—always press forward [$\models_{al\_press\_for}$] into possibilities [π]?

This sentence can be interpreted by the following system of formulas:

(8.1) $(((\varepsilon_{dim}(\alpha \subset_{discl} \mathfrak{U}) \models_{all\_press\_for} \pi) \models \mathfrak{U}) \models_{quest} \mathfrak{W}_{why};$
$((\alpha \models \varepsilon_{dim}) \models \alpha) \models \varepsilon_{dim}(\alpha);$
$(\mathfrak{U} \subset_{discl} \varepsilon_{dim}(\alpha)) \subset_{discl} \mathfrak{U}$

This formal system may appear to be stronger than the original sentence, but, it expresses the regular circular power of entities in question. The reader can try to think through this particular situation by himself. □

[8.2] It is because [$\Rightarrow \varphi_{(8.1)}$] the understanding has in itself the existential structure [$\sigma_{exist} \subset \mathfrak{U}$] we call "projection" [$\pi_{project}$].

A simple formula, modeling this sentence, is, for instance,

(8.2) $((\sigma_{exist} \subset \mathfrak{U}) \models \pi_{project}) \Rightarrow \varphi_{(8.1)}$ □

[8.3] With equal primordiality [$\models_{with\_eq\_p}$] the understanding projects Dasein's Being both upon [$\models_{proj\_upon}$] its "for-the-sake-of-which" [$\varphi_{sake}(\mathfrak{U})$] and upon significance [ξ_{sign}], as the worldhood [$\mathfrak{W}_{worldhood}$] of its current world [$\mathfrak{W}_{cur\_world}$].

One of formal possibilities for these sentence is, for instance,

(8.3) $(\mathfrak{U} \models_{with\_eq\_p} \mathfrak{B}(\mathfrak{D})) \models_{proj\_upon}$
$(\varphi_{sake}(\mathfrak{U}), \xi_{sign} \models_{as}$
$\mathfrak{W}_{worldhood}(\mathfrak{W}_{cur\_world}(\mathfrak{U}))$ □

[8.4] The character of understanding [$\gamma_{char}(\mathfrak{U})$] as projection [$\pi_{project}$] is constitutive [$\models_{const}$] for Being-in-the-world [$\mathfrak{B}_{in\text{-}the\text{-}world}$] with regard to [$\models_{with\_regard}$] the disclosedness [$\vartheta_{discl}$] of its existentially constitutive [$\models_{exist\_const}$] state-of-Being [$\mathfrak{S}_{of\text{-}Being}$] by which the factical potentiality-for-Being [$\varphi_{fact}(\pi_{for\text{-}Being})$] gets [$\models_{get}$] its leeway [$\lambda_{leeway}(\gamma_{char}(\mathfrak{U}))$] [Spielraum].

For this sentence, one of possible formalizations is

(8.4) $((\gamma_{char}(\mathfrak{U}) \models_{as} \pi_{project}) \models_{const}$
$\mathfrak{B}_{in\_the\text{-}world}) \models_{with\_regard}$
$\vartheta_{discl}((\mathfrak{S}_{of\text{-}Being}(\gamma_{char}(\mathfrak{U}))) \models_{exist\_const});$
$(\gamma_{char}(\mathfrak{U}) \models_{by} \varphi_{fact}(\pi_{for\text{-}Being})) \models_{get}$
$\lambda_{leeway}(\gamma_{char}(\mathfrak{U}))$

This formula system can be substantially modified by reading of the two equivalent German sentences, which are: Der Entwurfcharakter des Verstehens konstituiert das In-der-Welt-sein hinsichtlich der Erschlossenheit seines Da als Da eines Seinkönnens. Der Entwurf ist die existenziale Seinsverfassung des Spielraums des faktischen Seinkönnens. The modified system for these German sentences is

(8.4') $(\gamma_{char}(\pi_{project}(\mathfrak{U})) \models_{const} \mathfrak{B}_{in\text{-}the\text{-}world})$
$\models_{with\_regard}$
$(\vartheta_{discl}(\tau_{there}(\gamma_{char}(\pi_{project}(\mathfrak{U})))) \models_{as}$
$\tau_{there}(\pi_{for\text{-}Being}));$
$\pi_{project} \models_{exist}$
$\mathfrak{S}_{of\text{-}Being}(\lambda_{leeway}(\varphi_{fact}(\pi_{for\text{-}Being})))$ □

[8.5] And as thrown [τ_{thrown}], Dasein is thrown [$\models_{thrown}$] into the kind of Being [$\mathfrak{R}(\mathfrak{B})$] which we call "projecting" [$\mathfrak{P}_{project}$].

The formula is

(8.5) $(\mathfrak{D} \models_{as} \tau_{thrown}) \models_{thrown}$
$(\mathfrak{R}(\mathfrak{B}) \models \mathfrak{P}_{project})$ □

[8.6] Projecting [$\mathfrak{P}_{project}$] has nothing to do with [$\not\models$] comporting [$\mathfrak{C}_{comport}$] oneself [$\omega_{oneself}$] towards a plan [π_{plan}] that has been thought out [$\models_{think\_out}$], and in accordance with [$=\!\!|_{accord}$] which Dasein arranges [$\models_{arr}$] its Being.

The approximate formula for this sentence can be put as

(8.6) $\mathfrak{P}_{project} \not\models (\mathfrak{C}_{comport}(\omega_{oneself}) \models_{towards}$
$((\models_{think\_out} \pi_{plan})$
$=\!\!|_{accord} (\mathfrak{D} \models_{arr} \mathfrak{B}(\mathfrak{D}))))$ □

[8.7] On the contrary [$\models_{contra}$], any Dasein has, as Dasein, already projected itself; and as long [$\models_{as\_long\_as}$] as it is, it is projecting [$\mathfrak{P}_{project}$].

The formula system for this sentence is characteristically circular, that is,

(8.7) $\mathfrak{D} \models_{contra} ((\mathfrak{D} \models_{as} \mathfrak{D}) \models_{project} \mathfrak{D});$
$((\mathfrak{D} \models; \models \mathfrak{D}) \models_{as\_long\_as} \mathfrak{D}) \models \mathfrak{P}_{project}$ □

[8.8] As long as [$\models_{as\_long\_as}$] it is, Dasein always has understood [$\models_{al\_underst}$] itself and always will understand itself in terms of possibilities.

Cyclicity of Dasein and its possibilities is characteristic for this sentence which can be formally interpreted as

(8.8) $(((\mathfrak{D} \models; \models \mathfrak{D}) \models_{as\_long\_as} \mathfrak{D}) \models_{al\_underst}$
$\mathfrak{D}) \models_{as} \pi(\mathfrak{D})$ □

[8.9] Furthermore, the character of understanding [$\Upsilon_{char}(\mathfrak{U})$] as projection [$\pi_{project}$] is such [$\alpha$] that the understanding does not grasp thematically [$\models_{grasp\_thema}$] that upon which [α] it projects—that is to say, possibilities [π].

There is,

(8.9) $(\Upsilon_{char}(\mathfrak{U}) \models \pi_{project}) \models \alpha;$
$\mathfrak{U} \not\models_{grasp\_thema} ((\mathfrak{U}(\alpha) \models_{project} \alpha), \pi)$ □

[8.10] Grasping it [$\mathfrak{G}_{grasp}(\mathfrak{U})$] in such [$\varphi_{(8.9)}$] a manner [$\models_{in\_manner}$] would take away from [$\models_{take\_away}$] what is projected [$\mathfrak{P}_{project}(\alpha)$] its very character as a possibility [$\pi(\Upsilon_{char}(\mathfrak{U}))$], and would reduce [$\models_{reduce}$] it to [$\models_{to}$] the given contents which we have in mind [$\Upsilon_{content}(\mu_{mind})$]; whereas projection [$\pi_{project}$], in throwing [$\mathfrak{T}_{throw}$], throws before [$\models_{throw\_before}$] itself the possibility [π] as possibility, and lets it *be* as such.

This sentence delivers a complex formal interpretation, for instance,

(8.10) $(\mathfrak{G}_{grasp}(\mathfrak{U}) \models_{in\_manner} \varphi_{(8.9)}) \models_{take\_away}$
$((\mathfrak{U} \models \mathfrak{P}_{project}(\alpha)) \models_{as} \pi(\Upsilon_{char}(\mathfrak{U})));$
$\mathfrak{G}_{grasp}(\mathfrak{U}) \models_{reduce} (\mathfrak{U} \models_{to} \Upsilon_{content}(\mu_{mind}));$
$((\pi_{project} \subset \mathfrak{T}_{throw}) \models_{throw\_before}$
$\pi_{project}) \models$
$((\pi \models_{as} \pi) \models; \models (\pi \models_{as} \pi))$ □

[8.11] As projecting [$\mathfrak{P}_{project}$], understanding is the kind of Being of Dasein [$\mathfrak{R}(\mathfrak{B}(\mathfrak{D}))$] in which it *is* its possibilities [$\pi(\mathfrak{U})$] as possibilities.

Thus, for the last sentence of this paragraph, there is, formally,

(8.11) $(\mathfrak{U} \models (\pi(\mathfrak{U}) \models_{as} \pi(\mathfrak{U}))) \subset$
$((\mathfrak{U} \models_{as} \mathfrak{P}_{project}) \models \mathfrak{R}(\mathfrak{B}(\mathfrak{D})))$ □

2.9. *THE NINTH PARAGRAPH OF § 31 [BT]*

[9.1] Because [$\Rightarrow$] of the kind of Being [$\mathfrak{R}(\mathfrak{B})$] which is constituted [$\models_{const}$] by the *existentiale* of projection [$\varepsilon_{exist}(\pi_{project})$], Dasein is constantly 'more' [$\mathfrak{D} \models_{constantly} \mu_{more}(\mathfrak{D})$] than [$\models_{than}$] it factually [$\models_{factually}$] is, supposing [$\Rightarrow$] that one [$o_{one}$] might want to make [$\models_{make}$] an inventory of it [$\iota_{inventory}(\mathfrak{D})$] as [$\models_{as}$] something-at-hand [$\alpha_{at\text{-}hand}$] and list [$\models_{list}$] the contents of its Being [$\Upsilon_{content}(\mathfrak{B}(\mathfrak{D}))$], and supposing that one were able [$\models_{able}$] to do so.

The approximate formalization of this sentence might be a single implicative formula

(9.1) $(o_{one} \models_{able}$
$((o_{one} \models_{make} (\iota_{inventory}(\mathfrak{D}) \models_{as}$
$\alpha_{at\_hand})),$
$(o_{one} \models_{list} \Upsilon_{content}(\mathfrak{B}(\mathfrak{D}))))) \Rightarrow$
$((\varepsilon_{exist}(\pi_{project}) \models_{const} \mathfrak{R}(\mathfrak{B})) \Rightarrow$
$((\mathfrak{D} \models_{constantly} \mu_{more}(\mathfrak{D})) \models_{than}$
$(\mathfrak{D} \models_{factually}; \models_{factually} \mathfrak{D}))$ □

[9.2] But [$\Leftarrow$] Dasein is never more than [$\not\models_{more\_than}$] it factically is [$\models_{factically}$], for to its facticity [$\varphi_{fact}(\mathfrak{D})$] its potentiality-for-Being [$\pi_{for\text{-}Being}(\mathfrak{D})$] belongs essentially [$\in_{essen}$].

One of the implicative formulas for this sentence is

(9.2) $(\mathfrak{D} \not\models_{more\_than} (\mathfrak{D} \models_{factically})) \Leftarrow$
$(\pi_{for\text{-}Being}(\mathfrak{D}) \in_{essen} \varphi_{fact}(\mathfrak{D}))$ □

[9.3] Yet [$\models_{yet}$] as Being-possible [$\mathfrak{B}_{possible}$], moreover, Dasein is never anything [α] less [$\not\models_{less}$]; that is to say [in parallel], it is existentially [$\models_{exist}$] that which in its potentiality-for-Being [$\pi_{for\text{-}Being}(\mathfrak{D})$], it is *not yet* [$\not\models_{yet}$].

A formal depiction of this sentence could be, for instance,

(9.3) $((\mathfrak{D} \models_{as} \mathfrak{B}_{possible}) \not\models_{less} \alpha) \models_{yet} \mathfrak{D};$
$(\mathfrak{D} \models_{exist} \alpha) \models_{in} (\mathfrak{D} \not\models_{yet} \pi_{for\text{-}Being}(\mathfrak{D}))$

We can certainly bring up some slightly modified formulas for this sentence through the displacement of several operators. □

[9.4] Only because [$\models_{only\_because}$] the Being of the "there" [$\mathfrak{B}(\tau_{there})$] receives its Constitution [γ_{const}] through [$\models_{through}$] understanding and through the character of understanding [$\gamma_{char}(\mathfrak{U})$] as projection [$\pi_{project}$], only because it *is* what it becomes (or, alternatively, does not become), can it say to itself 'Become what you are', and say this with [$\models_{say\_with}$] understanding.

This sentence is rather complicated as it can be seen from the following formal example of it:

(9.4) $((\mathfrak{B}(\tau_{there}) \models_{say} \mathfrak{B}(\tau_{there}))$
$\models_{say\_with} \mathfrak{U}(\mathfrak{B}(\tau_{there}))) \models_{only\_because}$
$((\mathfrak{B}(\tau_{there}) \models_{receive} \gamma_{const}(\mathfrak{B}(\tau_{there})))$
$\models_{through} ((\mathfrak{U}; \gamma_{char}(\mathfrak{U}) \models_{as} \pi_{project});$
$(\mathfrak{B}(\tau_{there}) \models; \models \mathfrak{B}(\tau_{there});$
$\not\models \mathfrak{B}(\tau_{there}); \mathfrak{B}(\tau_{there}) \not\models)))$

The construction of this formula begins from the last part of the sentence, for sentence as a whole expresses an implicative intention [operator $\models_{only\_because}$]. Several explanations to formula (9.4) can be given which can substantially impact the understanding of the original sentence in a circular spontaneous manner and, simultaneously, stressing the way of Being of understanding in question together with the possibilities of its informational formal expression. For instance, the syntagma *it* is *what it becomes (or alternatively does not become)* concerning the Being of the "there", in the last formula is expressed in a completely and not only metaphysically open form, that is

$\mathfrak{B}(\tau_{there}) \models; \models \mathfrak{B}(\tau_{there});$
$\not\models \mathfrak{B}(\tau_{there}); \mathfrak{B}(\tau_{there}) \not\models$

where even the alternative general operator of non-informing $\not\models$ instead of $\not\models$ is used. The reader may imagine some further implications in possibilities of using such spontaneous circular mechanisms in an informational machine. □

2.10. *THE TENTH PARAGRAPH OF § 31 [BT]*

[10.1] Projection [$\pi_{project}$] always pertains to [$\models_{always} \circ \models_{pertain}$] the full disclosedness [$\varphi_{full}(\vartheta_{disclose})$] of Being-in-the-world [$\mathfrak{B}_{in\text{-}the\text{-}world}$]; as potentiality-for-Being [$\pi_{for\text{-}Being}$], understanding has itself possibilities which are sketched out beforehand [$\models_{sketch\_out\_beforehand}$] within the range [$\rho_{range}$] of what is essentially disclosable [$\models_{essen} \circ \models_{discl}$] in it.

This sentence can be reasonably decomposed in three consequent formulas:

(10.1) $\pi_{project} \models_{always} \circ \models_{pertain}$
$\varphi_{full}(\vartheta_{disclose}(\mathfrak{B}_{in\text{-}the\text{-}world}));$
$\pi(\mathfrak{U}) \subset (\mathfrak{U} \models_{as} \pi_{for\text{-}Being}));$
$\pi(\mathfrak{U}) \models_{sketch\_out\_beforehand} (\pi(\mathfrak{U}) \subset$
$((\rho_{range} \models_{essen} \circ \models_{discl} \mathfrak{W}_{what})$
$\models_{essen} \circ \models_{discl} \rho_{range}))$

In this system we introduced two composed types of operators between operands, that is, operator composition $\models_{always} \circ \models_{pertain}$ and $\models_{essen} \circ \models_{discl}$. This does not mean that in these compositions the left operator pertains merely to the left operand and the right operator merely to the right operand. Such explicit operator composition pertains always to both operands. Operators, possessing transitive function, can be understood as distributed operational units, interwoven with operands. □

[10.2] Understanding *can* devote itself primarily [$\models_{devote}$] to [$\models_{prim\_to}$] the disclosedness of the world [$\vartheta_{disclose}(\mathfrak{W}_{world})$]; that is, Dasein can, proximally [$\models_{prox}$] and for the most part [$\models_{most\_part}$], understand [$\models_{\mathfrak{U}}$] itself in terms of its world [$\mathfrak{W}_{world}(\mathfrak{D})$].

Using operator composition of the form $\models_{prox} \circ \models_{most\_part}$, the last sentence can be formalized in the following way:

(10.2) $(\mathfrak{U} \models_{devote} \mathfrak{U}) \models_{prim\_to} \vartheta_{disclose}(\mathfrak{W}_{world});$
$(\mathfrak{D} \models_{\mathfrak{U}} \mathfrak{D}) \models_{prox} \circ \models_{most\_part} \mathfrak{W}_{world}(\mathfrak{D})$

As one can see, the function of operators $\models_{devote}$ and $\models_{\mathfrak{U}}$ becomes transitive in regard to operator $\models_{prim\_to}$ and to operator composition $\models_{prox} \circ \models_{most\_part}$, respectively. □

[10.3] Or else [in parallel] understanding throws [$\models_{throw}$] itself primarily into [$\models_{prim\_into}$] the "for-the-sake-of-which" [φ_{sake}]; that is, Dasein exists as [$\models_{exist\_as}$] itself.

With the transitive function of operator $\models_{throw}$ in regard to operator $\models_{prim\_into}$ one can formalize this sentence as

(10.3) $(\mathfrak{U} \models_{throw} \mathfrak{U}) \models_{prim\_into} \varphi_{sake}$;
$\mathfrak{D} \models_{exist\_as} \mathfrak{D}$ □

[10.4] Understanding is either [$\models_{either}$] authentic [$\mathfrak{U}_{auth}$], arising out of [$\models_{arise\_out}$] one's own Self [$\sigma_{self}(o_{one})$] as such, or inauthentic [$\mathfrak{U}_{inauth}$].

One of possible formulas for this sentence is

(10.4) $(\mathfrak{U} \models_{either} (\mathfrak{U}_{auth} \models_{arise\_out}$
$(\sigma_{self}(o_{one}) \models_{as} \sigma_{self}))) \models_{or}$
$(\mathfrak{U} \models \mathfrak{U}_{inauth})$ □

[10.5] The 'in-' [ι_{in}] of "inauthentic" [ι_{inauth}] does not mean [$\not\models_{mean}$] that Dasein cuts itself off [$\models_{cut\_off}$] from its Self [$\sigma_{self}(\mathfrak{D})$] and understands 'only' [$\models_{\mathfrak{U}} \circ \models_{only}$] the world.

There is, formally,

(10.5) $(\iota_{in} \models_{of} \iota_{inauth}) \not\models_{mean}$
$((\mathfrak{D} \models_{cut\_off} \mathfrak{D}) \models_{from} \sigma_{self}(\mathfrak{D}))$;
$\iota_{inauth} \not\models_{mean} (\mathfrak{D} \models_{\mathfrak{U}} \circ \models_{only} \mathfrak{W}_{world})$ □

[10.6] The world belongs to [$\in$] Being-one's-Self [$\mathfrak{B}_{one's\text{-}self}$] as Being-in-the-world.

A simple formalization of this sentence is

(10.6) $(\mathfrak{W}_{world} \models_{as} \mathfrak{B}_{in\text{-}the\text{-}world}) \in \mathfrak{B}_{one's\text{-}Self}$ □

[10.7] On the other hand [in parallel], authentic understanding [$\mathfrak{U}_{auth}$], no less than that which is inauthentic [$\mathfrak{U}_{inauth}$], *can* be either genuine [one can choose $\gamma_{genuine}$] or not genuine.

A further way of formalization of authentic and inauthentic understanding is, according to the last sentence,

(10.7) $(\mathfrak{U}_{auth}, \mathfrak{U}_{inauth} \models_{either} \gamma_{genuine}) \models_{or}$
$(\mathfrak{U}_{auth}, \mathfrak{U}_{inauth} \not\models \gamma_{genuine})$ □

[10.8] As potentiality-for-Being [$\pi_{for\text{-}Being}$], understanding is altogether permeated [$\models_{permeate}$] with possibility.

There is

(10.8) $\pi \models_{permeate} (\mathfrak{U} \models_{as} \pi_{for\text{-}Being})$ □

[10.9] When one [o_{one}] is diverted [$\models_{divert}$] into [Sichverlegen in] one of these basic possibilities [π_{basic_1}] of understanding, the other [π_{basic_2}] is not laid aside [$\models_{lay\_aside}$] [legt ... nicht ab].

We separately marked the one and the other basic possibility, thus,

(10.9) $(\pi_{basic\_1}(\mathfrak{U}) \models_{divert} o_{one}) \Rightarrow$
$(\not\models_{lay\_aside} \pi_{basic\_2}(\mathfrak{U}))$

This formula is a temporal implication which pertains to the form *when ... then ...* of the last sentence. □

[10.10] *Because understanding, in every case, pertains rather to* [$\models_{always} \circ \models_{pertain}$] *Dasein's full disclosedness* [$\varphi_{full}(\vartheta_{disclose}(\mathfrak{D}))$] *as Being-in-the-world, this diversion* [$\vartheta_{diversion}$] *of the understanding is an existential* [$\models_{exist}$] *modification of projection* [$\mu_{modif}(\pi_{project})$] *as a whole* [$\mathfrak{W}_{whole}$].

One possible formal interpretation of this sentence is

(10.10) $((\mathfrak{U} \models_{as} \mathfrak{B}_{in\text{-}the\text{-}world}) \models_{always} \circ \models_{pertain}$
$\varphi_{full}(\vartheta_{disclose}(\mathfrak{D}))) \Rightarrow$
$((\vartheta_{diversion}(\mathfrak{U}) \models_{as} \mathfrak{W}_{whole}) \models_{exist}$
$\mu_{modif}(\pi_{project}))$

For the right side of this implication we could put also

$\vartheta_{diversion}(\mathfrak{U}) \models_{exist}$
$(\mu_{modif}(\pi_{project}) \models_{as} \mathfrak{W}_{whole})$

depending of our understanding of the sentence. □

[10.11] In understanding the world [$\mathfrak{U}(\mathfrak{W}_{world})$], Being-in is always understood [$\models_{always} \circ \models_{\mathfrak{U}}$] along with it [$\mathfrak{B}_{in}(\mathfrak{U}(\mathfrak{W}_{world}))$], while [in parallel] understanding of existence [$\mathfrak{U}(\varepsilon_{ex})$] as such is always an understanding of the world.

Formalization of this sentence is, for instance,

(10.11) $\mathfrak{U}(\mathfrak{W}_{world}) \models_{always} \circ \models_{\mathfrak{U}} \mathfrak{B}_{in}(\mathfrak{U}(\mathfrak{W}_{world}))$;
$(\mathfrak{U}(\varepsilon_{ex}) \models_{as} \mathfrak{U}(\varepsilon_{ex})) \models_{always} \mathfrak{U}(\mathfrak{W}_{world})$

The while in the last sentence is interpreted by the parallelism of formulas, however, it could also be expressed explicitly introducing a particular operator. □

2.11. *THE ELEVENTH PARAGRAPH OF § 31 [BT]*

[11.1] As factical Dasein [$\mathfrak{D}_{fact}$], any Dasein has already diverted [$\models_{divert}$] its potentiality-for-Being into [$\models_{into}$] a possibility of understanding.

A straightforward formula for this sentence is

(11.1) $(\mathfrak{D} \models_{as} \mathfrak{D}_{fact}) \models_{divert}$
$(\pi_{for\text{-}Being}(\mathfrak{D}) \models_{into} \pi(\mathfrak{U}))$

In this formula, the diverting (operator $\models_{divert}$) means *'diverts'* as *'has already diverted'*. □

2.12. *THE TWELFTH PARAGRAPH OF § 31 [BT]*

[12.1] In [$\models_{in}$] its projective character [$\gamma_{char}(\pi_{project}(\mathfrak{U}))$], understanding goes to make up [$\models_{make\_up}$] existentially [$\models_{exist}$] what we call Dasein's *"sight"* [$\sigma_{sight}(\mathfrak{D})$] [*Sicht*].

We interpret this sentence by formula

(12.1) $\gamma_{char}(\pi_{project}(\mathfrak{U})) \models_{in}$
$(\mathfrak{U} \models_{make\_up} \circ \models_{exist} \sigma_{sight}(\mathfrak{D}))$

The 'in' at the beginning of the sentence has the meaning 'inform(s) in' (operator $\models_{in}$). □

[12.2] With the disclosedness of the "there" [$\vartheta_{disclose}(\tau_{there})$], this sight [$\sigma_{sight}(\mathfrak{D})$] is existentially [$\models_{exist}$] [existenzial seiende]; and Dasein *is* this sight equiprimordially in each of those basic ways [$\models_{equi\_p} \circ \models_{always}$] of its Being which we have already noted: as the circumspection [$\gamma_{circumsp}$] [Umsicht] of concern [$\gamma_{concern}$], as the considerateness [$\gamma_{consider}$] [Rücksicht] of solicitude [$\pi_{solicitude}$], and as that sight [σ_{sight}] which is directed upon Being as such [$\mathfrak{B} \models_{as} \mathfrak{B}$] [Sicht auf das Sein als solches], for the sake of which [$\models_{sake}$] any Dasein is as it is [$\mathfrak{D} \models; \models \mathfrak{D}$].

We can formally interpret this sentence by a system of two formulas:

(12.2) $\sigma_{sight}(\mathfrak{D}) \models_{with} (\vartheta_{disclose}(\tau_{there}) \models_{exist})$;
$((\mathfrak{D} \models \sigma_{sight}(\mathfrak{D})) \models_{equi\_p} \circ \models_{always}$
$\mathfrak{B}(\mathfrak{D})) \models_{as}$
$(\gamma_{circumsp}(\gamma_{concern}), \gamma_{consider}(\pi_{solicitude}),$
$\sigma_{sight}((\mathfrak{B} \models_{as} \mathfrak{B}) \models_{sake} (\mathfrak{D} \models; \models \mathfrak{D})))$

In this case, the meaning of operator $\models_{exist}$ is 'inform(s) existentially'. □

[12.3] The sight [$\sigma_{sight}(\mathfrak{D})$] which is related primarily and on the whole to [$\models_{rel\_prim\_whole}$] existence [ε_{ex}] we call *"transparency"* [$\tau_{transpar}$] [*Durchsichtigkeit*].

There is

(12.3) $(\varepsilon_{ex} \models_{rel\_prim\_whole} \sigma_{sight}(\mathfrak{D})) \models \tau_{transpar}$

In this formula, operator $\models_{rel\_prim\_whole}$ is a composition of operators $\models_{relate}$, $\models_{primarily}$, and $\models_{on\_the\_whole}$, that is, for instance,

$(\models_{relate} \circ \models_{primarily}) \circ \models_{on\_the\_whole}$ □

[12.4] We choose this term [$\tau_{transpar}$] to designate 'knowledge of the Self' [$\mathfrak{K}_{know}(\sigma_{self})$] in a sense which is well understood [$\sigma_{sense}(\mathfrak{U}_{well})$], so as to indicate [in parallel] that here it is not [$\not\models$] a matter of perceptually tracking down and inspecting a point called the "Self" [$\mathfrak{P}_{percept}(\sigma_{self})$], but [$\models_{but}$] rather one of seizing [$\mathfrak{S}_{seize}$] upon the full disclosedness of Being-in-the-world [$\varphi_{full}(\vartheta_{disclose}(\mathfrak{B}_{in\text{-}the\text{-}world}))$] *throughout all* [$\models_{throughout} \circ \models_{all}$] the constitutive items [$\iota_{item}(\gamma_{const})$] which are essential to it, and doing so with understanding.

The formula system for this sentence could be

(12.4) $\tau_{transpar} \models$
$(\mathfrak{K}_{know}(\sigma_{self}) \models \sigma_{sense}(\mathfrak{U}_{well}))$;
$(\tau_{transpar} \not\models \mathfrak{P}_{percept}(\sigma_{self}) \models_{but}$
$(\mathfrak{S}_{seize}(\varphi_{full}(\vartheta_{disclose}(\mathfrak{B}_{in\text{-}the\text{-}world})))$
$\models_{throughout} \circ \models_{all}$
$(\iota_{item}(\gamma_{const}) \models_{essen} (\tau_{transpar}, \mathfrak{U})))$

This formula is an evident case of formal abstraction and modification (symbolic simplification) of

the original sentence. However, it is still open in many respects, so additional improvements and even universalization of it can be easily performed. The reader can expect some help in understanding the German sentence which is the following: Wir wählen diesen Terminus zur Bezeichnung der wohlverstandenen »Selbsterkenntnis«, um anzuzeigen, daß es sich bei ihr nicht um das wahrnehmende Aufspüren und Beschauen eines Selbstpunktes handelt, sondern ein verstehendes Ergreifen der vollen Erschlossenheit des In-der-Welt-seins *durch* seine wesenhaften Verfassungsmomente *hindurch*. We agree that this sentence can deliver an informationally entirely different formalization to formula (12.4). □

[12.5] In existing [$\mathfrak{E}_{exist}$], entities [α] sight [$\models_{sight}$] 'themselves' [sichtet "sich" only in so far as [$\models_{so\_far}$] they have become transparent to themselves [$\tau_{transpar}(\alpha)$] with equal [$\models_{with} \circ \models_{equal}$] primordiality [$\pi_{prim}$] in those items [$\iota_{item}$] which are constitutive for their existence [$\mathfrak{E}_{exist}(\alpha)$]: their Being-alongside the world [$\mathfrak{B}_{alongside}(\mathfrak{W}_{world})$] and their Being-with Others [$\mathfrak{B}_{with}(\omega)$].

For the last sentence of this paragraph we have the following formalization:

(12.5) $(\alpha \models_{sight} \mathfrak{E}_{exist}(\alpha)) \models_{so\_far}$
$(\alpha \models (\tau_{transpar}(\alpha) \models_{with} \circ \models_{equal}$
$\pi_{prim}(\iota_{item} \models \mathfrak{E}_{exist}(\alpha)))$;
$\mathfrak{E}_{exist}(\alpha) \models \mathfrak{B}_{alongside}(\mathfrak{W}_{world}, \mathfrak{B}_{with}(\omega))$ □

2.13. *THE THIRTEENTH PARAGRAPH OF § 31 [BT]*

[13.1] On the other hand [in parallel], Dasein's opaqueness [$\mathfrak{D}_{opaque}(\mathfrak{D})$] [Undurchsichtigkeit] is not rooted [$\not\models_{root}$] primarily and solely in [$\models_{prim\_sol}$] 'egocentric' [ε_{ego}] self-deceptions [$\mathfrak{D} \models_{decept} \mathfrak{D}$]; it is rooted [$\models_{root}$] just as much in [$\models_{just\_as\_much}$] lack of acquaintance [$\mathfrak{A}_{acquaint}$] with the world.

For the only sentence of this paragraph we have

(13.1) $\mathfrak{D}_{opaque}(\mathfrak{D}) \not\models_{root} \circ \models_{prim\_sol}$
$\varepsilon_{ego}(\mathfrak{D} \models_{decept} \mathfrak{D})$;
$\mathfrak{D}_{opaque} \models_{root} \circ \models_{just\_as\_much}$
$\lambda_{lack}(\mathfrak{A}_{acquaint} \models_{with} \mathfrak{W}_{world})$

There are certainly other possibilities for the formalization of Dasein's egocentric self-deceptions. □

2.14. *THE FOURTEENTH PARAGRAPH OF § 31 [BT]*

[14.1] We [one] must, to be sure [$\models_{sure}$], guard against [$\models_{guard\_against}$] a misunderstanding [$\mathfrak{U}_{mis}$] of the expression 'sight' [$\varepsilon_{express}(\sigma_{sight})$].

The formula we can propose is

(14.1) $(o_{one} \models_{guard\_against} \mathfrak{U}_{mis}(\varepsilon_{express}(\sigma_{sight})))$
$\Rightarrow (o_{one} \models_{sure})$ □

[14.2] It [$\varepsilon_{express}(\sigma_{sight})$] corresponds [$\models_{corresp}$] to the "clearedness" [$\gamma_{clear}$] [Gelichtetheit] which we took as characterizing [$\models_{char}$] the disclosedness of the "there" [$\vartheta_{disclose}(\tau_{there})$].

There is, for instance,

(14.2) $\varepsilon_{express}(\sigma_{sight}) \models_{corresp}$
$(\gamma_{cleared} \models_{char} \vartheta_{disclose}(\tau_{there}))$ □

[14.3] 'Seeing' [$\mathfrak{S}_{see}$] does not mean just [$\not\models_{mean} \circ \models_{just}$] perceiving [$\mathfrak{P}_{perceive}$] with the bodily eyes [$\varepsilon_{eyes}(\beta_{body})$], but [in parallel] neither does it mean pure non-sensory [$\models_{pure} \circ \not\models_{sense}$] awareness of something [$\mathfrak{A}_{aware}(\alpha)$] present-at-hand [$\alpha_{present\_at\_hand}$] in its presence-at-hand [$\pi_{at\_hand}(\alpha)$].

The less or more appropriate formula system for this sentence is

(14.3) $\mathfrak{S}_{see} \not\models_{mean} \circ \models_{just} (\mathfrak{P}_{perceive} \models_{with}$
$\varepsilon_{eyes}(\beta_{body}))$;
$\mathfrak{S}_{see} \not\models_{mean}$
$(\mathfrak{A}_{aware}(\alpha) \models_{pure} \circ \not\models_{sense}$
$((\alpha \models \alpha_{present\_at\_hand}) \subset \pi_{at\text{-}hand}(\alpha)))$
□

[14.4] In giving [$o_{one} \models_{give}$] an existential signification [σ_{sign_exist}] to "sight" [σ_{sight}], we have merely drawn upon [$\models_{draw\_upon} \circ \models_{merely}$] the peculiar feature of seeing [$\varphi_{feature}(\pi_{peculiar}(\mathfrak{S}_{see}))$], that it lets entities [α] which are accessible [$\models_{access}$] to it be encountered unconcealedly [$\models_{encounter} \circ \not\models_{conceal}$] in themselves.

Formulas corresponding to this sentence are

(14.4) $(o_{one} \models_{give} \sigma_{exist}(\sigma_{sight}))$
$\models_{draw\_upon} \circ \models_{merely}$
$\varphi_{feature}(\pi_{peculiar}(\mathfrak{S}_{see}))$;
$\alpha \models_{encounter} \circ \not\models_{conceal}$
$(\alpha \models_{access} \varphi_{feature}(\pi_{peculiar}(\mathfrak{S}_{see}))) \square$

[14.5] Of course, every 'sense' [σ_{sense}] does [$\models_{do}$] this [$\varphi_{(14.4)}$] within that domain of discovery which is genuinely its own [$\vartheta_{domain}(\vartheta_{discover}(\sigma_{sense}))$].

A simple formal interpretation of this sentence is

(14.5) $(\sigma_{sense} \models_{do} \varphi_{(14.4)} \subset$
$\vartheta_{domain}(\vartheta_{disclose}(\sigma_{sense})) \square$

[14.6] But from the beginning onwards the tradition of philosophy [$\beta_{begin}(\tau_{trad}(\pi_{philo}))$] has been oriented primarily towards [$\models_{orient} \circ (\models_{prim} \circ \models_{towards})$] 'seeing' as a way of access [$\mathfrak{W}_{way}(\mathfrak{A}_{access})$] to entities *and to Being.*

A formula for this sentence is

(14.6) $\beta_{begin}(\tau_{trad}(\pi_{philo}))$
$\models_{orient} \circ (\models_{prim} \circ \models_{towards})$
$(\mathfrak{S}_{see} \models_{as} (\mathfrak{W}_{way}(\mathfrak{A}_{access}) \models_{to} \alpha, \mathfrak{B})) \square$

[14.7] To keep the connection with [$\models_{connect}$] this tradition, we [o_{one}] may formalize [$\models_{formalize}$] "sight" and "seeing" enough to obtain [$\models_{enough} \circ \models_{obtain}$] therewith a universal term [τ_{term_uni}] for characterizing any access to entities or to Being, as access in general [$\mathfrak{S}_{general}$].

For the last sentence of this paragraph there is

(14.7) $\tau_{trad} \models_{connect}$
$(o_{one} \models_{formalize}$
$(\sigma_{sight}, \mathfrak{S}_{see} \models_{enough} \circ \models_{obtain}$
$(\mathfrak{S}_{term\_uni} \models_{char}$
$(\mathfrak{A}_{access} \models_{to} (\alpha, \mathfrak{B} \models_{as}$
$(\mathfrak{A}_{access} \subset \mathfrak{S}_{general}))))))) \square$

2.15. *THE FIFTEENTH PARAGRAPH OF § 31 [BT]*

[15.1] By showing [$\models_{show}$] how all sight is grounded primarily in [$\subset_{ground} \circ \subset_{prim}$] understanding (the circumspection of concern [$\gamma_{circumsp}(\gamma_{concern})$] is understanding as *common sense* [$\gamma_{common}(\sigma_{sense})$] [*Verständigkeit*]), [$\Rightarrow$] we have deprived [$\models_{deprive}$] pure intuition [$\pi_{pure}(\iota_{intuition})$] [Anschauen] of its priority [π_{prior}], which corresponds noetically to [$\models_{corresp} \circ \models_{noe}$] the priority of the present-at-hand in traditional ontology [$\tau_{trad}(o_{ontology})$].

The implicative formula for this sentence is

(15.1) $(o_{one} \models_{show}$
$((\sigma_{sight} \subset_{ground} \circ \subset_{prim} \mathfrak{U}),$
$(\gamma_{circumsp}(\gamma_{concern}) \models$
$(\mathfrak{U} \models_{as} \gamma_{comm}(\sigma_{sense})))) \Rightarrow$
$((o_{one} \models_{deprive} (\pi_{pure}(\iota_{intuition} \models_{of}$
$\pi_{prior}(\pi_{pure}(\iota_{intuition})))) \models_{corresp} \circ \models_{noe}$
$\pi_{prior}(\pi_{at\_hand} \subset \tau_{trad}(o_{ontology})) \square$

[15.2] 'Intuition' and 'thinking' [$\mathfrak{T}_{think}$] are both derivatives of understanding [$\vartheta_{derive\_remote}$], and already rather remote ones.

A simple formula is

(15.2) $\iota_{intuition}, \mathfrak{T}_{think} \models_{already}$
$\vartheta_{derive\_remote}(\mathfrak{U}) \square$

[15.3] Even the phenomenological [$\varphi_{phenomenal}$] 'intuition of essences' [$\iota_{intuition}(\varepsilon_{essence})$] ["Wesenschau"] is grounded in [$\subset_{ground}$] existential understanding [$\mathfrak{U}_{exist}$].

An adequate formula for this sentence is

(15.3) $\varphi_{phenomenal}(\iota_{intuition}(\varepsilon_{essence}))$
$\subset_{ground} \mathfrak{U}_{exist} \square$

[15.4] We can decide about [$\models_{decide}$] this kind of seeing [$\Re(\mathfrak{S}_{see})$] only if we have obtained explicit conceptions of Being [$\gamma_{concept\_ex}(\mathfrak{B})$] and of the structure of Being [$\sigma_{structure}(\mathfrak{B})$], such as only phenomena in the phenomenological sense can become [$\lambda_{logic}(\varphi_{phenomenal})$].

The implicative case of formalization of this sentence is

(15.4) $(\gamma_{concept\_ex}(\mathfrak{B}), \sigma_{structure}(\mathfrak{B}) \models_{as}$
$\lambda_{logic}(\varphi_{phenomenal})) \Rightarrow$
$(o_{one} \models_{decide} \Re(\mathfrak{S}_{see}(\mathfrak{B})))$

The last formula is a formal generalization which pertains to the entire realm of the Heideggerian Being. $\square$

(Will be continued)

RRL - AN INTEGRATED ENVIRONMENT OF ROBOT PROGRAMMING

INFORMATICA 1/92

Keywords: robot programming, off-line programming, robot controller, CAD systems

Bojan Nemec[1], Anton Ružič[1], Vinko Ilc[2]
[1]Jozef stefan Institute, University of Ljubljana
[2]Riko - Ribnica

The paper describes RRL, an integrated robot programming environment, which includes a robot controller with a dialogue-oriented robot monitor, editor, robot language interpreter and an additional computer for graphic simulation and off-line program development. Trajectories can be generated by a CAD system, verified on the off-line programming system and then transferred and executed on the real robot. One of the main advantages of the system is that it can be implemented on a low cost-hardware. Off-line programming uses the identical trajectory generation module as the robot controller and includes simulation of the robot dynamics. Therefore, the simulation gives extremely credible results. The system described is in use in several industrial applications.

RRL - integrirano okolje za programiranje robotov:
Članek opisuje programski sistem RRL, ki vključuje programsko opremo na robotskem krmilniku in programsko opremo na zunanjem računalniku za programiranje z simulacijo. Trajektorije, ki sestavljajo delavno nalogo, robota lahko določimo s pomočjo CAD sistema, preiskusimo z simulacijo in prenesemo v robotski krmilnik. Prednost predstavljenega sistema sistema je tudi v tem, da je implementiran tudi na cenenih PC računalnikih.

1 Introduction

Unlike in NC machine programming, there is no industrial standard for robot programming. Almost each robot manufacturer offers his own robot programming language [1,2,3]. Textual robot programming languages offer many advantages, but the teach-in principle is still widely used for robot programming [1,2,3]. The main reasons why the teach-in principe is still used are the following:

- simple operation; the operator does not have to learn any programming language.
- teaching by showing nearly eliminates all logical errors in trajectory definition.
- simple debugging and correction of the robot program.

debugging and correction of robot programs can be efficiently done by off-line development and verification via graphic simulation. Additionally, off-line programming supported by CAD allows work cell analysis in the early stage of work cell planning [4]. In the past, many off-line programming systems were developed [5,6,7,8], but there are some problems which limit the use of off-line robot programming, such as:

- Post processors are not available only for all robots or robot programming languages.
- If the simulation software does not include the same type of trajectory generation algorithms, the simulation will give false results, especially in cycle time calculation.
- Most of the off-line simulation packages do not include the dynamics of the robot, hence it is always questionable if the simulation will be close enough to the behaviour of the real robot, especially at high speed.

- off-line programming often requires expensive hardware and software which may surpass the cost of the robot itself.

For the new generation of RIKO industrial robots, we developed a programming environment called RRL (Riko Robot Language) which tries to solve the above-mentioned problems. The RRL program environment has the versatility of textual programming languages and ease of use and debugging capabilities of teach-in programming. RRL allows off-line program development and graphic simulation which runs on a low cost PC computer, coupled with a robot controller.

2 Basic structure of RRL

The RRL robot program environment consists of two main modules:

- RRL program environment on the robot controller, which allows on-line program development and execution.
- RRL program environment on the host computer, which allows off-line program development and simulation.

2.1 RRL program environment on the robot controller

RRL on the robot controller consists of the following modules (Fig 1) :

- PasRo kernel. Pasro is a set of pascal procedures and functions for robot programming [9]. It is a powerful robot programming language by itself, but requires knowledge of Pascal programming and a Pascal compiler to run. In our case, PasRo is used as a kernel for RRL development.
- RRL interpreter is the interpreter of a high level, motion-oriented robot programming language.
- RRL editor is a menu-oriented, screen editor. It is designed to simplify the writing and editing of the textual part of the RRL program. A syntax check is performed at this level and therefore it is impossible to write a syntactically incorrect program.
- RRL frames definition module consists of a frame editor where frames are entered or edited explicitly using the main control panel and the Teach-in module, where frames are defined implicitly using the teach panel.
- RRL monitor, which links other modules and allows saving and loading of robot programs. Up to 42 programs can be stored the RRL directory. There are several modes of execution of the robot program :
  - continuous execution of the program,
  - step-by-step execution,
  - continuation of execution of the program. The program can be changed and then continued from every step.
  - step-by-step execution of the last 32 movements in the reverse order.
- RRL computer link module allows connection of the robot controller to the host computer. The host computer can send or receive any program or parameter field from or to the robot controller. Additionally, it can take control of the execution of the program and send commands directly to the robot or receive data from the robot. The entire programming system is menu-oriented. The user does not have to type or remember all commands but simply selects the required command from the menu.

2.2 Robot language description

The RRL is designed as a compromise between simplicity of use and ability to program complex tasks. The main demands in developing RRL were:

- to be simple to understand and simple to learn,
- to allow easy menu oriented programming,
- to be easily expandable with new instructions.

RRL can drive a robot with up to 6 robot axes and up to 3 external axes and can coordinate robot trajectory with up to 3 arbitrarily positioned external axes. RRL can be used to program most of the

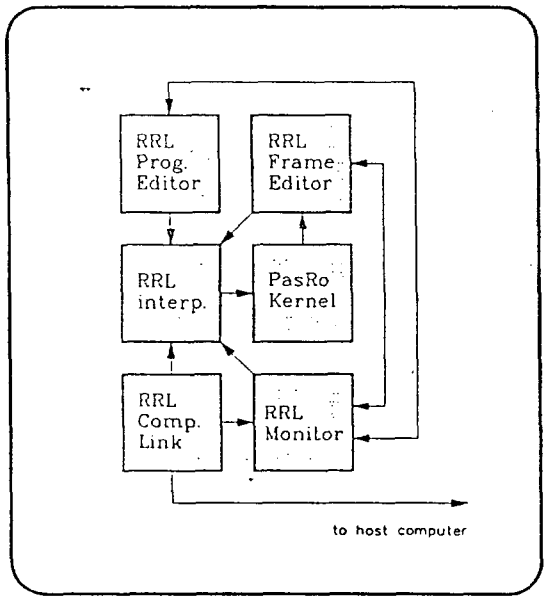


Figure 1: Structure of the RRL system and data flow on the robot controller

tasks which can be performed by the RIKO 106 welding/assembly robot. For very complex tasks with intensive sensor interaction, we can develop a program on the host computer in PasRo and control the robot with the host computer through the Computer Link.

RRL contains instructions for :

- different types of interpolations

JMOVE	movement with joint interpolation
SMOVE	straight move in Cartesian coordinates
VMOVE	SMOVE via intermediate frames without stopping
CIRCLE	circular interpolation
SPLINE	spline interpolation through a set of frames
LINE	straight move with selectable velocity profile
APPRO	approach to the point with joint interpolation
DEPART	straight relative movement
DRIVE	relative movements of the joint
FTRACK	activating of hybrid force position control using up to 6 d.o.f force/torque sensor.

All movement instructions except LINE generates a smooth trajectory using 4-1-4 splines [10].

- instructions for movement definition

MAXSP	definition of the max. speed in mm/s and dg/sec
SPEED	relative speed in % of maximum speed
SPPR	selection of the speed priority
MOVSEL,	selection and definition of the coordination
SYNMO	between robot axes and external axes
ACC	selection of the acceleration factor
TCP	tool geometry definition. Up to 10 tools with tcp vector and orientation angles can be defined
SHIFT	on line shift and rotation of the frames
PALLET	definition for manipulation with up to 10 pallets

- frame arithmetics

FRAME	assignment, addition, subtraction and rotation of the frames.

- instruction for general program facilities

SET	assignment to the integer value
GOTO, CALL	unconditional branching and calling a subprogram
IF THEN ENDIF	conditional branching
WRITE	output on the user window of the screen
CHAIN	chain to another robot program
DELAY, STOP	

- instructions for process synchronization

IFS THEN ENDIF	conditional branching on input signal status
SIGON, SIGOF, PULSE	activating of an output signal
WAIT	wait for the defined event

- instructions for arc welding

WLDST, WLDSP	start and stop of arc welding
WLDCL	cleaning of weldin pistol
WEAVE	weaving with different patterns during arc welding
WPAR	definition of welding parameters

2.3 RRL programing environment for the host computer

The basic structure of the RRL off-line programming environment is presented in Fig 2. and consists of the following modules :

- PasRo kernel, which is identical to the kernel of the RRL kernel for the robot controller
- RRL interpreter, which is identical to the interpreter of the robot controller
- RRL program file and location definition file compiler, which compiles the source code for the textual program and file definition into the internal format of the robot controller.
- Align module, which translates and rotates the whole location definition file to the correct position, using three reference points
- Any text editor
- CAD package with the capability of defining 3-D wireframe objects (e.g. AutoCad Version 9.0 or later)
- Simulation system module, which simulates the kinematics and dynamics of the robot and displays the simulated cell on the graphic terminal. Although the simulation module is primarily intended for the Riko-106 6 axis electrical robot, it can be user-adapted to other robots with similar kinematic configurations.
- Module for converting DXF drawing format to the internal format of the simulation system. The user can define or modify the robot or cell components using a CAD package. DXF file format is then converted to another format for faster 3-D animation.
- Module for automatic generation of RRL programs using a CAD system. The user can define the robot trajectory in AutoCad on the selected layer. This module generates the RRL program for the defined trajectory.

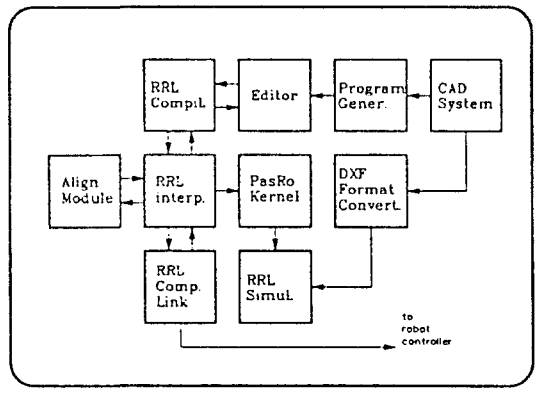


Figure 2: Structure and data flow of the RRL off-line system

3 Hardware implementation

3.1 RRL On-Line programming system

The RRL on-line programming system is implemented on the robot controller, based on a VME bus. The structure of the robot controller is presented in Fig 3. The organization of the controller hardware is hierarchical. The RRL system is implemented on the main CPU under O.S.-9. The main CPU features a Motorola 68020 with a Motorola 68881 arithmetical coprocessor for fast trajectory calculation and 1 Mb of RAM. The axis CPU is another Motorola 68010, where a digital servo controller with feed-forward compensation of velocity errors and PLC (programmable logic controller) controllers are implemented. All peripheral equipment such as D/A, A/D, incremental encoders, digital I/O are attached to the Motorola I/O bus. With such an organization, access to the peripherals equipment does not overload the VME bus, which is primarily intended for processor communication. Battery back-up RAM is used as an external memory device for saving application programs.

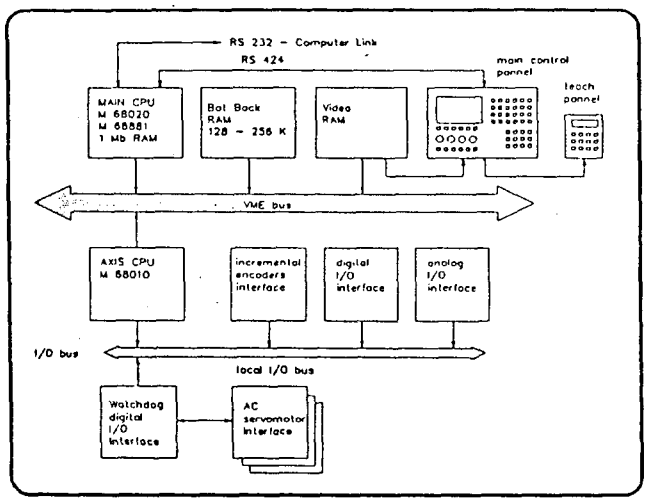


Figure 3: Block diagram of the robot controller

3.2 RRL Off-Line programming system

The RRL off-line programming system is implemented on a personal computer with MS-DOS O.S. and VAX or MicroVAX with VMS O.S. PC implementation offers wireframe models and wireframe representation without hidden line removal. In order to achieve reasonable simulation speed, PC implementation includes simplified dynamic models of the robot and actuators. PC implementation allows simulation of one 6. d.o.f robots, one object with 6. d.o.f and object with fixed coordinates during simulation. The simulation speed is determined by the complexity of the modelled environment. A typical configuration for PC includes a PC-AT 16 Mhz computer with arithmetical coprocessor, an EGA or VGA display and a display adapter. With such hardware, the simulation speed is about 3 times slower than execution of the same program on the real robot. VMS implementation is enhanced by interfacing to ROMAS - Robot Cell Modelling and Simulation System. The functions of ROMAS can be grouped into the following four modules:

- solid modeller for defining complex solids by translation, rotation and union of primitive solids;
- kinematic modeller for defining general kinematic chains having rotational, translational and parallelogram joints. Using this module we can define robots, grippers, positioners etc.
- cell modeller for definition of cell layouts using previously defined components and describing their spatial relationships and their connection types.
- robot cell simulation using a meta-language with commands for positional control of cell components, for defining changes in spatial and connection relationships, etc.

During simulation, ROMAS calculates the effects of program execution on the cell structure and generates the animated scene on the graphic terminal. When used with 3D graphics terminals (e.g. Tektronix 4236), it generates hidden the lines or shaded display on-line. With its capabilities, ROMAS represents a general tool for choosing appropriate robots for specific tasks, evaluating alternative cell layouts and developing logically and positional correct program structures.

We have connected RRL and ROMAS to overcome the limitations of each system. Specifically, we can describe robot tasks in RRL using the target robot language and calculate the target robot dynamics. On the other hand, using ROMAS, we can quickly define complex cells, simulate simultaneous control of different robots and machines and evaluate different programs and cell layouts.

4 Example program

A sample RRL application program is shown in Figure 4. The task is to find the welding gap of the part and weld the part with weaving. For the sake of simplicity, the position of the welding gap is assumed to be unknown only in one direction and a tactile sensor is used to detect the edge, although the more general case can be easily programed using frame arithmetics.

```
* sample program
   maxsp = 1000 ; maximal speed [mm/s]
   speed = 50   ; relative speed [%]
   tcp 1 = 0 240 0 0 0 0 ; tool centre
   set y = 20   ; variable for searching
* find edge of the part
   appro to 1 for 0 -10 1
   call find
   frame 1 = frame 0 ; current robot frame
   appro to 2 for 0 -10 1
   call find
   frame 2 = frame 0
* execute welding
```

```
   appro to 1 for 0 5 5
   smove to 1
   speed = 2.5
   wave 2 5 2 0
   line to 2
   depart for 0 5 5
   speed = 50
   home
   stop
* subprogram for detecting the edge
* of the part with tactile sensor.
* Tactile sensor activates signal 4
   label find
   depart for 0 y 0 until sig 4 hi
   if sig 4 lo then
         write text Cant'Find
         stop
   endif
   return
```

Figure 4: Sample RRL program

Fig. 5 shows the graphic output of the simulation of the above program on a PC computer. Fig. 6 shows the graphic output of the simulation of the pick-and-place task with coordination with the CNC machine on a VAX computer with Tektronix 3236 graphics terminal.

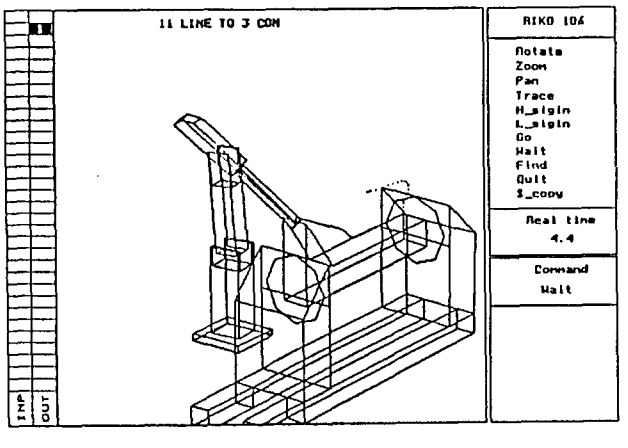


Figure 5: Examples of the graphic output of the simulation on the PC computer

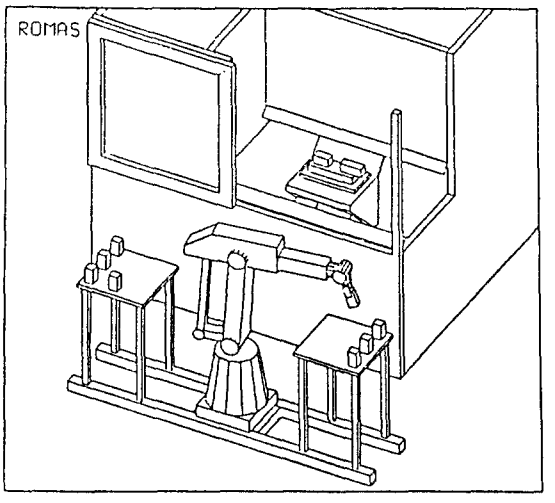


Figure 6: Example of the graphic output of the simulation on the VAX computer

5 Conclusion

RRL was demonstrated to be a successful, versatile and easy-to-learn robot programming system in many industrial applications. In addition, the RRL off-line module is an excellent tool for training and education. The RRL programming system was primary developed for the RIKO 106 electrical robot, intended for arc welding and assembly tasks. Due to the modular design of the system, it can be easily adapted for use in other robots. For a similar kinematic configuration, the system can be user-adapted by changing system parameters. For a completely different robot, only the kinematic transformation module has to be changed. Up to now, RRL was also installed on a 5-axis Cartesian robot for grinding plastic casts.

6 References

1. Bonner S., Shin K.G.: A Comparative Study of Robot Languages, IEEE Computer, Dec. 1982

2. Gruver W.A., Soroka B.I., Craig J.J, Turner T.T.: Industrial Robot Programming Languages: A Comparative Evaluation, IEEE Vol SMC-14, No4, 1984

3. Lozano-Perez T. : Robot Programming, IEEE Vol. 71. No. 7, 1982

4. Worn H., Stark G.: Robot Applications

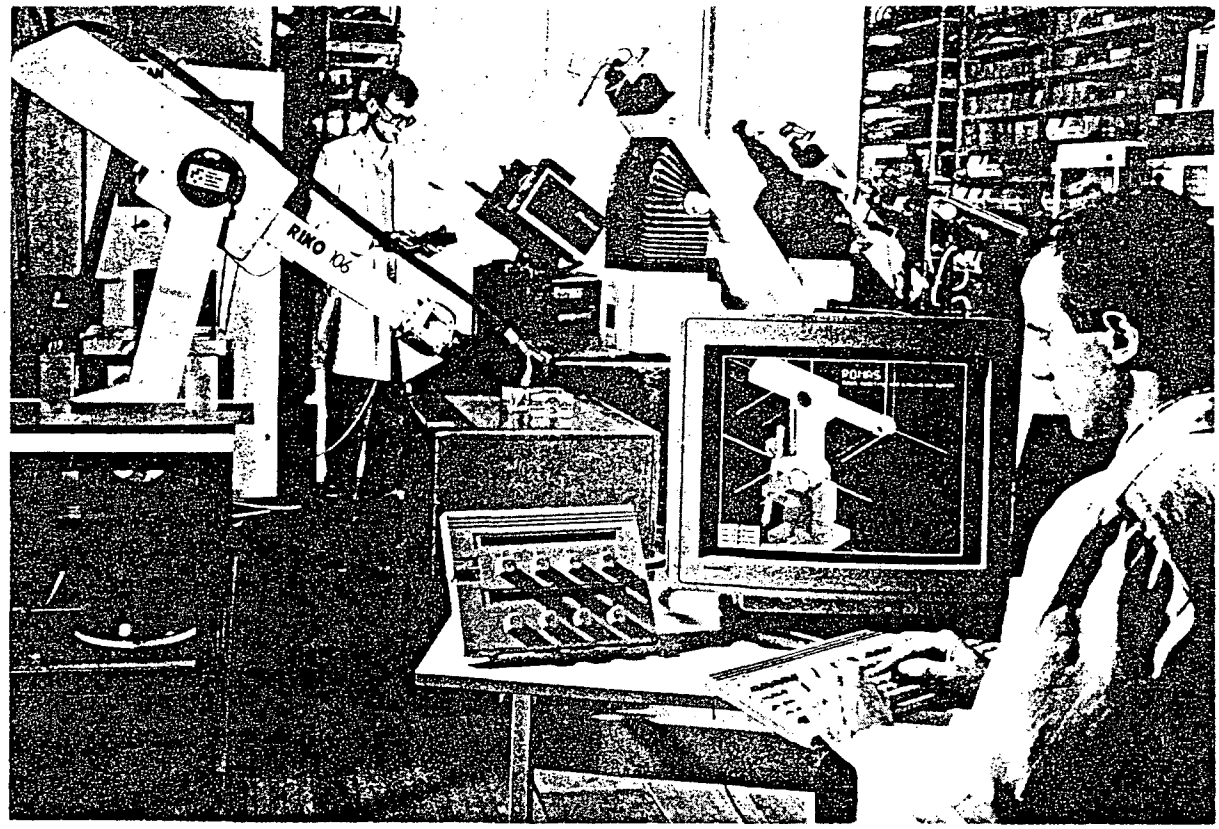


Figure 7: Riko 106 welding/assembly robot with robot controller and VAX based off-line programming system.

Supported by CAD Simulation, Robotics and Computer-Integrated Manufacturing, Vol3, No. 1, 1987

5. Milberg J., Schufer N., Tauber A.: Requirements for Adavanced Robot Programming Systems, symp SyRoCo'88, Karlsruhe, 1988

6. Nemec B., Lenarcic J.: A Robot Simulation System Based on Kinematic Analyses, Robotica, Vol. 3, 1985

7. Dombre E., Borrel P., Liegeois A.: A CAD System for Programming and Simulation Robots' Action, Digital Systems for Industrial Automation, Vol. 2, No. 2, 1984

8. Sol E.J., van der Broek A. Th.M : Progress in CAD-Tools for Robot Based Flexible Automation Systems, 14th ISIR, Gotenburg, 1984

9. Blume C., Jakob W.: Programming Languages for Industrial Robots, Springer-Verlag, 1986

10 Paul R.P.: Robot Manipulators : Mathematics, Programming, and Control, MIT Press, Cambridge, 1981

Keywords: progress of computers, microelectronics, storage media, parallel computing, multimedia, personal computers

Matjaž Gams[1], Borut Hribovšek[1,2]
[1]Institut »Jožef Stefan«, Ljubljana
[2]ISKRA Elektrooptika, Ljubljana

ABSTRACT: In "Trends of Computer Progress" (Gams, Žitnik 1990) basic trends of computer progress were presented as an answer to growing speculations of either stagnation or spectacular breakthroughs. More detailed and technically supported survey is given in this second part, starting from microelectronics, storage media, parallel and distributed computing to operating systems and software technology, communications, multimedia, and finally PC trends. The overview strongly indicates that in the next 10 years progress will continue with similar astonishing pace as it did for the last 50 years (Baldi 1991).

POVZETEK: V članku "Trends of Computer Progress" (Gams, Žitnik 1990) so bile nakazane osnovne smeri razvoja računalnikov kot odgovor na naraščujoče špekulacije ali o stagnaciji ali o še bolj spektakularnem prodoru. V tem drugem delu so podane bolj natančne in tehnično podprte smernice razvoja mikroelektronike, pomnilniških medijev, paralelnega in porazdeljenega procesiranja, operacijskih sistemov, programske opreme, multimedijev in osebnih računalnikov. Pregled potrjuje zaključek, da se bo v naslednjih 10 letih razvoj nadaljeval z nezmanjšano hitrostjo (Baldi 1991).

1 Microelectronics

In the last 30 years [1], since the industrial application took place, the progress of microelectronics is the essence of computer progress. An overview shows the following major indicators:

- Performance (density, speed) steadily increases while the cost per bit steadily decreases.
- Applications (penetration in new products) steadily increase.
- For the nest 10 years, the mainstream technology is expected to be CMOS since it seems the best suited for VLSI or ULSI devices.

1.1 Performances

Integration density.
According to the Moore's law (Moore 1975), the number of transistors per chip has increased exponentially and has so far doubled every 1.5 years. The law can be observed from two lines in Figure 1 (source Intel) where the right line represents the progress in logical devices while the left one represents the progress in memory. The increase has been faster for memories due to more regular and simpler structure while logic devices require longer design times due to more complex functions.

Speed.
Gate speed constantly increases and already reaches 0.1 ns. However, while gate delay progressively decreases, circuit access time does not increase proportionally due to more and more complicated circuits.

Size.
Growing density and speed are based on the reduction in feature size and the increase of die size (Figure 2, source Intel). The growth was enabled by the ongoing progress in lithography and the improvements in the quality of materials.

[1]This survey is based on several magazines such as Byte, Future Generation Computer Systems, AI Magazine, etc.

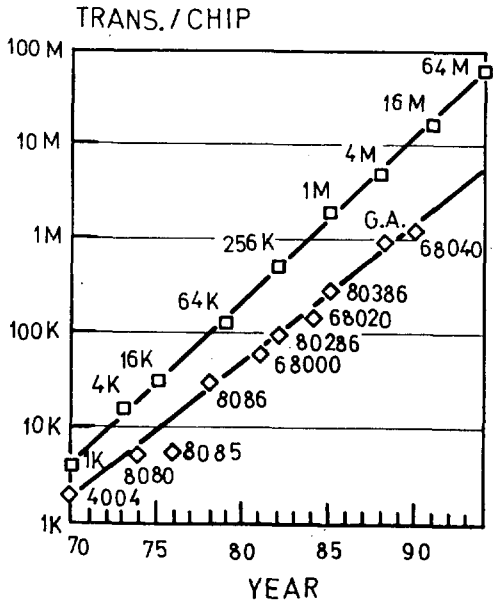


Figure 1: Plotting the number of transistors per chip versus time shows the Moore's law: every 1.5 years the integration density doubles.

1.2 Costs

In the last twenty years, the cost per bit exponentially decreases (Figure 3, source I.C.E. Report 1990). However, while the overall cost per bit decreases, process cost (i.e. production or equipment cost) actually increases and represents one of the biggest annoyances in future progress.

1.3 Applications

In recent years, microelectronics has become a key component of electronic equipment. Microelectron-

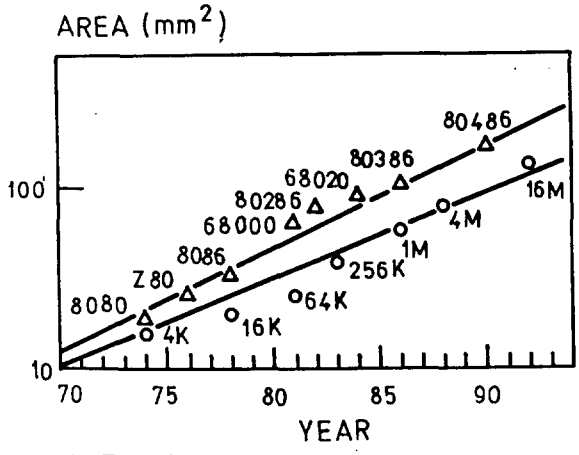


Figure 2: Die chip size versus time shows constant improvements.

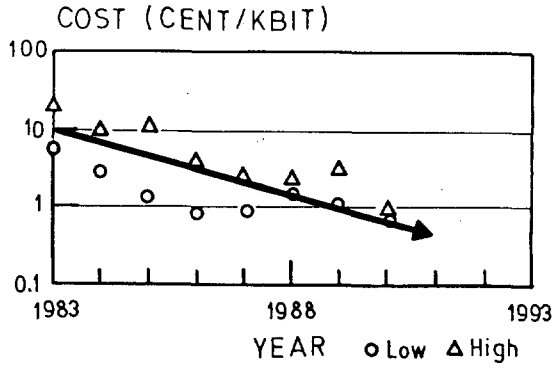


Figure 3: Cost per bit versus time for DRAM (Dynamic Random Access Memory).

ics have gained important new areas like videorecorders and CD's, while there is a constant growth in already established areas like computer applications. The ratio between semiconductors and equipment sales is presented in Figure 4 (I.C.E. Report 1990). The crossing point is near year 2150, when nearly all technical devices will be equipped by semiconductors. This broader use brings development of new features like analog circuitry or digitally controlled power capabilities and will effect every-day life in more or less every human activity, e.g. in smart houses.

1.4 Main technologies

Today, three main technologies are present on the market: bipolar, MOS (both based on silicon), and GaAs. The main technologies seem to be more complementary than competitive, each of them presenting a preferred field of application.

MOS (Metal-Oxide Semiconductor) technology consists of

- PMOS (Positive-well MOS) - obsolete,
- NMOS (Negative-well MOS) - being phased out,
- CMOS (Complementary MOS) - mainstream technology for VLSI, and

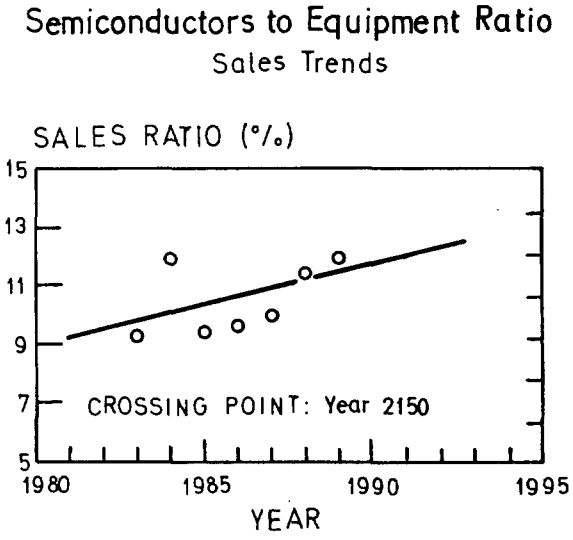


Figure 4: Ratio between semiconductors and equipment sales indicates an ongoing progress in semiconductor applications.

- BiCMOS (a hybrid of Bipolar and CMOS technologies) - in development.

CMOS devices are voltage-driven with relatively high threshold voltage, high input impedance and relatively low current drive capability. The main advantages are: low complexity, high input impedance and low power dissipation, which make them especially suited for large scale integration and modular circuit design. The drawbacks are low current driving capabilities, and relatively low speed.

Bipolar technology consists of

- ECL (Emitter Coupled Logic) - fastest silicon based process in growth,
- TTL (Transistor-Transistor Logic) - main bipolar logic technology phasing out, and
- LINEAR - mainstream analog technology in competition for complex devices.

Bipolar devices are current controlled devices with low threshold voltage, low input impedance and high current driving capabilities. The biggest advantages are: high current drive capability and good analog performances, which make them ideally suited for analog devices and for high speed logic. The most important drawbacks are high power dissipation, process complexity and low input impedance, which do not allow modular design and the use for large scale integration.

GaAs devices

- OPTO - well defined market, expected to grow steadily,
- LOGIC - high costs confine it to very special applications

make use of several basic transistor structures. However, due to several problems the integration density is still by several orders of magnitude lower than for silicon based devices and the costs are much higher. On the other hand, the carrier mobility is more than 5 times faster than in silicon and the bandgap can be directly tailored. This makes GaAs ideally suited for optoelectronic and very high speed digital and analog applications, fields in which device cost is not a critical issue.

At present, MOS devices account for more than 50% of semiconductor's market (Figure 5, I.C.E. Report 1990) with GaAs covering only a meager 0.5%. In the next years, the total disappearance of PMOS and NMOS devices is expected. The share of CMOS and GaAs devices is further expected to increase.

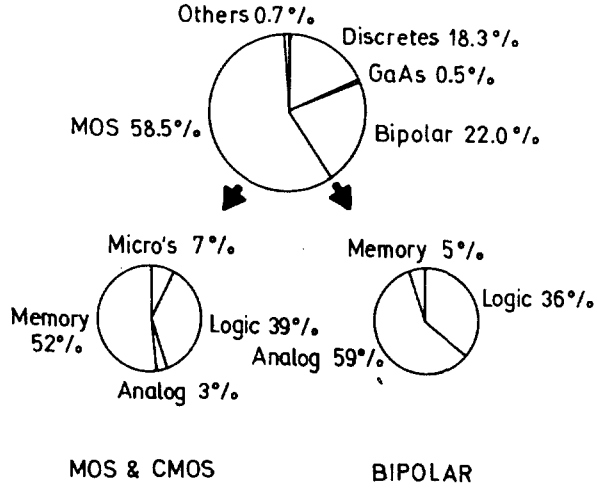


Figure 5: Split of market share among different technologies.

1.5 Present status and technological limits

Gate delays are in the 100-200 ps range for CMOS, and down to 25 ps for bipolar ECL technology.

While general extrapolation shows no major obstacle for further development, the following critical issues remain to be:

- lithography
  The minimal feature size has been reduced from more than 10 μm to less than 1 μm in the last twenty years. The progress has been achieved through advances in optical lithography and, most importantly, no real barrier is likely to appear in the next ten years.

- transistor architecture
  The first critical point has been reached for MOS transistors with device lengths around 1.2 μm due to the increase in power density. While present solutions might go to 0.6 - 0.7 μm, new solutions can be expected at the level of the scaling rules and the reduction of the supply voltage around 3 - 3.5 V.

- interconnections
  Since logic devices are becoming more and more complex, a certain limit of interconnections is becoming one of crucial factors. The solution seems to be the use of more metal levels, going from present two levels to three or more.

- defect density
  In the last twenty years, the particle density in the production environment has been reduced by at least three orders of magnitude. While the production costs grow, the problem of defect density basically translates to the cost limitations.

- cost limitations
  The cost of semiconductor facilities steadily grows and, perhaps, the darkest observation is that specific cost grow even faster. For example, the lithography costs versus feature size seem to be exponential.

1.6 The near future

Given the large amount of investments in the field, given the strong interconnections between microelectronics, the bulk of electronic industry, and the existing R&D prototypes, no revolutionary change will take place in the near future of around ten years. Performances, e.g., speed, capacity or complexity will continue to grow with exponential growth as they did so far. Technology will continue to change as rapid or even faster than today. Therefore, due to constant introducing of new approaching and the growth in performances, microelectronic industry seems to be a young industry. However, from the point of investment, growing manufacturing costs, and a decrease of the number of semiconductor companies, it appears a mature industry branch with all its pluses and minuses.

2 Storage media

On today's market, devices with 4 Mbit DRAM's and 1Mbit SRAM's (Static RAM) can be obtained, and 16 Mbit DRAM's exist as prototypes. 64 Mbit DRAM's are being developed in development laboratories and basic elements for the future 256 Mbit chip are being studied in research laboratories. Following this projection and the one in Figure 1, it seems reasonable to assume that 256 Mbit DRAM's with a 0.25 micron geometry will be on the market by the end of the century. Therefore, personal computer memories will reach 100 Mbytes, workstation memories will reach 1 Gbyte, while mainframes will offer from 10 up to 100 Gbytes. In terms of logic, personal computers will reach 100 MIPS, workstations around 1GIPS and mainframes from 10 GIPS up to 100 GIPS. The difference between workstations and personal computers will be more in terms of price and purpose than in technological advances.

Besides microelectronics, several other areas like magnetic storage technology record important progress as well. Performances, such as density, access speed and transfer rates continue to improve. By the end of the century, head-disk interface should go beyond one tenth of a micron and aerial densities should grow from the current 100.000 bits per mm^2 to one megabit per mm^2. With continued improvement in speed, capacity, and price/performance ratio, hard disk drives can and probably will still remain the preferred direct-access storage devices.

The biggest challenge to magnetic mass storage comes from optical technologies (Ryan 1990) such as CD-ROM, WORM (Write Once, Read Many times), and erasable optical disks. Optical storage is slower than magnetic primary because of the greater mass of optical read/write heads, while on the other hand it offers greater capacity. Quite

probably, optical storage will develop in parallel with magnetic storage and will be used in low-cost storage for low-end personal computers.

Therefore, the magnetic media hierarchy will basically remain unchanged in the years to come. The relationship between access and capacity is shown in Figure 6 (source Byte). The fastest technologies have the smallest capacity and the slowest technologies have the largest capacity. The pyramid makes a rough correlation between the height of each block and the percentage of each type of storage present in a typical system.

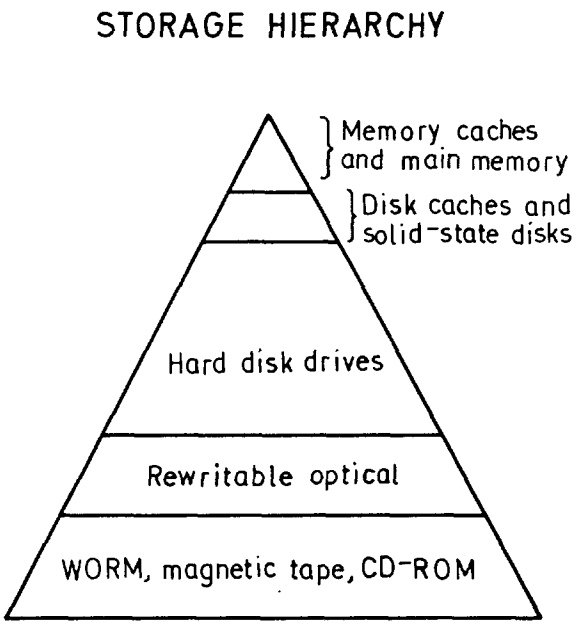


Figure 6: Storage hierarchy.

What impacts will these computer performances provide (Duby 1991)? Certainly, several tasks will be done in other way than today. For example, disk sorting techniques will be less important due to a simple fact that most of the sorting will be performed in the main memory. Simple calculation shows than 10 years from now even personal computers will have enough central memory to store (and sort) names and surnames of all people in Slovenia. Similarly, several hierarchical techniques will also be less often used. For example, relational databases will benefit from those memory sizes while hierarchical database management systems will be in decline. Large databases will influence every-day applications of artificial intelligence and knowledge-based systems.

3 Parallel and distributed computing

In next ten years, more and more attention will be devoted to parallel and distributed computing (Hertzberger 1991). At present, parallel computers with a large number of tightly coupled processors are commercially available and loosely coupled networks of computers are quite commonly used. However, classical Von Neumann computer architecture will probably remain dominant for at least 5-10 years since there are some difficult unsolved problems in parallel computing.

Some areas of parallel computing will evolve naturally with growing computer, microelectronic and transmission performances. For example, visualisation requires specialised processors and high speed communications requires quick accepting and storing large amounts of data. It is also quite likely that specialised artificial neural network processors will be used for pattern recognition tasks. Numerical intensive applications are another successful area for parallelism. Not surprisingly, since the parallelism in supercomputers, e.g., vectorization of applications, was an essential step to improve processing speed.

Different classes of parallelism are being identified:

- domain parallelism, where data structures are distributed among various processors,
- algorithmic parallelism, where the computer network is designed to match a particular algorithm and suitable code fragments and data flows are constructed, and
- task-parallelism, where a problem is divided into a large number of subproblems.

Until now, much less progress can be reported in general or even specialised symbolic application area. The largest experimental project where a large non-numeric application, i.e. a knowledge-base system, was the driving force behind the design of a parallel computer system, was the Japanese Fifth Generation Computer System. Results were mixed, with no breakthrough but with great impacts on future computer research and development. At present, several similar or competitive

projects run in Japan, Western Europe and USA. At first, logic programming language Prolog was utilised for implicit parallelism. In the second approach, programmers explicitly control parallelism usually by object oriented languages. What is used today is essentially an improvement of existing programming languages such as Modula or C++.

One of the fundamental problems is coordination and control of the communications among parallel fragments that comprise the task and one of the major bottlenecks in parallel computing is the lack of a coherent general model of describing and organising a parallel computing process. Lately, there were some promising attempts, unfortunately without greater practical meaning. In one approach (Valiant 1990), the possibility was shown of defining an idealised parallel processor PRAM, similar to the Turing's machine. It was also shown that under certain conditions, an algorithm runs n times faster on a parallel machine with n processors.

Therefore, at least theoretically it can be shown that parallel universality exists. On the other hand, practical solutions seem still quite far away as can be shown by a simple calculation: If only 1% of the whole process has to be solved in a sequential way, any parallel machine with any number of processors can not achieve an improvement by the factor 100.

Also, in the area of distributed computing some of the problems might slow down the expected progress, maybe simply by the unwillingness of many user communities to fully exploit the new possibilities by paying the price of forgetting the old-style approach and corresponding knowledge and skills. However, several implications seem inevitable, e.g., the role of supercomputers will quite probably decrease because of the cost-effectiveness of parallel and distributed computing.

4 Operating systems and software

In **operating systems**, less time will be devoted to memory allocation and more to new functions like distributed services and data, symbolic data query, cooperative processing and fault detection and recovery. After all, with 100 Mbytes on personal computers who needs virtual memory?

In the PC arena (Baran 1992), it is remarkable how little operating systems have changed since the introduction of the IBM PC and, a couple of years later, the Macintosh. With the exception of the Mac OS, which has had an integrated GUI since its introduction in 1984, the big change in the operating-system arena in recent years has been the addition of windowing systems and GUI's to DOS and Unix, both of which traditionally have had command-line interfaces.

Of course, there is one obvious reason for the slow change in the operating-system technology, and this is compatibility with the huge data and applications base that already exists on millions of computers today.

Software technology is getting more complicated (Nance 1992). Developers have to hack through a jungle of computer languages, operating environments, user interfaces, and shifting standards to choose how to create their software. Therefore, one of important guides is toward standardisation and manageability of different aspects of software.

One of the main improvements will be more and more complex applications. Simulated models will replace experiments and enable efficient search for optimal solutions in many areas. Very large and complex data will enable access of all interesting data such as encyclopaedic data bases or business history for specific branches. Also, multimedia will be part of every-day activities combining mass products like television and personal computers. In industrial applications, CIM (CAD, CAP, CAM, CAQ and PPC) will become one of the most common and widely used approaches.

5 Communications and multimedia

Progress in communication technologies is expected to grow even faster than in computer technologies. In a decade, bandwidth possibility is expected to reach a few megabit per second from existing kilobits per second. It will effect architecture as well as disk needs since new transmission techniques will enable gigabytes per second access rates.

Due to the availability of many different kinds of data, standardisation will become pervasive in about every domain of computer technologies and applications from hardware to software, communi-

cations and user interfaces. Yet, communication with computers will become much more human-like and application-user-oriented since most of the users will basically have no knowledge about programming.

Multimedia uses the computer to integrate and control diverse electronic media (Robinson 1990) such as computer screens, CD-ROM disks, videodisk players, speech and audio synthesizers. Multimedia definitions run from combining text, sound, and animation onscreen to full digital video for editing and storage.

User interface will be one of areas where additional speed and capacities will enable great improvements. High definition graphic interface for both input and output will become pervasive while speech will enable high quality output as well as reasonable input. Since around 1 GIPS is needed for high speech and graphics performance, workstations near the end of the century will be able to do it at professional level while low-price personal computers will have to be contented with reasonable compromises. Commercial TV uses approximately 230 million bits per second and high quality personal computer displays need up to 500 million bits per second. This is far more than available disks on PC's enable: hard magnetic disks have about 16 million bits per second while CD optical disks enable 1.5 million bits per second. At the moment, CD's seem more prospective since they are removable and already a consumer product. The second problem is the storage capacity which at the moment enables only around one tenth of a minute of video data. However, there are two bright points: first is the expected growth in computer performances and the second improvement can be expected in compression techniques. If the loss of some fidelity is accepted (lossy compression) then even today it is possible to achieve compression rates of 50:1 for still images and up to 200:1 for moving video.

Therefore, expensive personal computers and workstations will achieve "marriage" with TV in forthcoming years. On the other hand, several new peripheral devices will become available such as smart scanners, readers and general data collectors from instruments and sensors. Most important, these new products will be task-user oriented for specific profiles. Also, color printing will considerably develop. In display technologies, cathode ray tubes are expected to remain most widely used with evolutionary improved resolution.

6 PC world

Progress in personal computers is one of the most effective and, also, has strongly influenced all human activities. The first PC (Campbell 1990), as defined by IBM, processed approximately one-tenth of a MIPS (1 Million Instructions Per Second). In those days (the early 1980s), it costed about $50.000 to put a theoretical 1 MIPS of processing power on your desktop. Today, 40 MIPS PSs are present on the market, and 100 MIPS PC's should arrive around 1995. This improvement in collective processing power will bring the cost per MIPS down below $50 (Figure 7).

The nose-diving cost of raw computing power is the result of two factors: progress in microcomputer technology and the growing numbers of manufacturers of integrated system logic, graphics, I/O, and communications chip sets.

The early PC's were shipped with 4K-byte and then 16K-byte DRAM's. Today, PC's are routinely shipped with 8 Mbytes or more memory. Future machines will be designed to accept many megabytes of memory largely because today's application programs are starting to demand more memory space for data and for the programs themselves.

Also, significant trends in the PC world are: the drive for higher levels of integration, the sudden rise of alternative processors, multi-processor architecture and, also, several RISC-based (Reduced Instruction Set Computer) machines are making a play to become a factor in the PC world.

Despite enormous success, jumping up to the next level of personal computer performance may not happen smoothly. For example, the limit of around 50-100 MIPS seems quite a difficult one to overcome by existing PC technology. On the other hand, computers in general will tend to progress with similar speed as today and, quite probably, another technology will be introduced.

The explanation of lines in Figure 7 is as follows:
1. 8088/86 PC, 96 SSI chips, 4.77 to 8+MHz
2. 80286 PC, 4 VLSI, 40 SSI chips, 6 to 25 MHz
3. 80386 PC, 4 VLSI, 40 SSI chips, 16 to 25 MHz

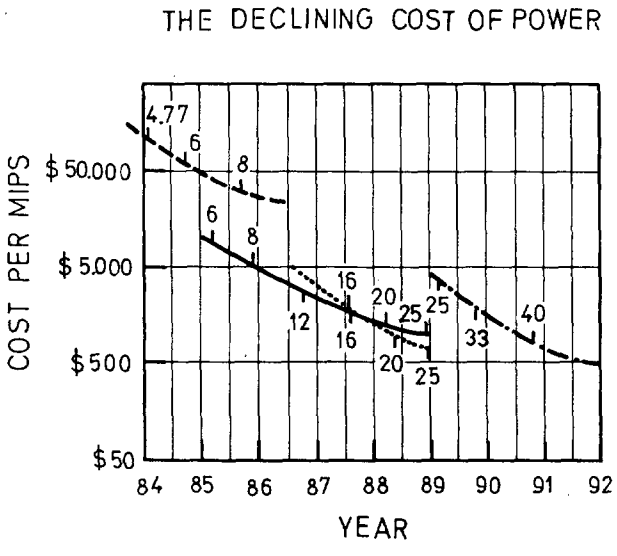


Figure 7: The declining cost of processing power.

4. 80386/486 PC (with cache), 3 VLSI, 19 SSI chips, 25 to 40+MHz

7 Summary

Exponential computer progress has been basically fuelled by the exponential growth in microelectronics. As a direct consequence, other computer-related activities progress with similar astonishing speed, among them storage media technology, parallel and distributed computing, software and operating systems, communications, multimedia, personal computers, databases, knowledge-based and artificial intelligence systems. This progress has been very constant over the last 50 years and was observed, for example, as the "Moore's law" in microelectronics. The scope of progress is expected to remain constant over the next 10 years and real technological limits are still far away.

Today, as a direct product of this astonishing development, a powerful PC on our desk has about 50 MIPS, 10 Mbytes of central memory and up to 1 Gbyte disk. Equipped with coprocessors and cache memory, it can compete with workstations designed a couple of years ago. Similarly, new powerful workstations approach performances of a couple of years old mainframe computers. For another comparison, today's PC's achieve performances of supercomputers designed 10 years ago. This means that const-effectiveness improves faster for smaller machines and, consequently, the market share will continue to improve for PC's and workstations while the share of supercomputers and mainframe computers will continue to decline. Especially, when having in mind the expected progress in parallel, distributed computing and high speed, high rate transmission networks.

References

Baldi L. (1991): Microelectronic trends, Future Generation Computer Systems 7.

Baran N. (1992): Operating systems now and beyond, Byte, Special issue, January.

Campbell G.A. (1990): Inventing the PC's future, Byte, Extra issue, January 20.

Duby J.J. (1991): The evolution of information technologies in the 90s and its impact on applications, Future Generation Computer Systems 7.

Gams M., Žitnik K. (1990): Trends of computer progress, Informatica 11.

Hertzberger L.O. (1991): Trends in parallel and distributed computing, Future Generation Computer Systems 7.

I.C.E. (1990): Mid-term 1990 status and forecast of the IC industry.

Moore G.E. (1975): Progress in digital integrated electronics, Technical Digest of 1975 International Electronic Devices Meeting 11.

Nance B. (1992): The future of software technology, Byte, Special issue, January 15.

Robinson P. (1990): The four multimedia gospels, Byte, February.

Ryan B. (1990): The once and future king, Byte, November.

Valiant L.G. (1990): General purpose parallel architecture, Handbook of Theoretical Computer Science, ed. Leeuween, North-Holland.

ON THE FUNCTIONALITY AND STRUCTURE OF INTELLIGENT INSTRUMENTATION SYSTEMS

INFORMATICA 1/92

Keywords: intelligent systems, knowledge representation, modelling, object-oriented design

Nikola Bogunović
Ruđer Bošković Institute
Zagreb

ABSTRACT: A rational approach to the functionality and structure of the intelligent instrumentation systems, dedicated to industrial process monitoring and control, and fault diagnosis applications, is presented. The functional model is distinguished from the structural model. Functions are represented by interrelated modules that support multiple levels of abstractions. Structural model is described through hierarchy of layered components. Particular attention is paid to process model building procedures, and to knowledge representation and reasoning techniques. While model building has settled around techniques that employ object-oriented and windowing methodology, knowledge representation procedures follow very diverse approaches.

KRATAK SADRŽAJ: U radu je analiziran djelatni i strukturni ustroj inteligentnih instrumentacijskih sustava namijenjenih za praćenje i vođenje industrijskih procesa, te za dijagnostiku kvarova. Razlučen je djelatni model od strukturnog modela. Funkcije sustava predstavljene su međusobno povezanim modulima u okviru zajedničke okosnice koja podržava višestruke razine apstrakcija. Strukturni model je opisan hijerarhijskim razinskim slojevima različitih komponenata. Posebna pažnja posvećena je procedurama tijekom izgradnje modela procesa, te tehnikama predstavljanja znanja i zaključivanja. Dok se u izgradnji modela gotovo u pravilu koriste grafički objektno orjentirani postupci, predstavljanje znanja slijedi vrlo različite pristupe.

INTRODUCTION

To maximise the information gain from measurement and control instrumentation systems, applied to physical and life sciences, medicine and engineering, the trend towards more complex instrumentation systems is observed. This is followed by the change in the architecture of contemporary instrumentation systems which permits implementation of complex and abstract models of applicable physical, chemical, and information transformation processes. In general, simple measuring devices, dedicated to measurement of specific physical quantities that can be described by a single scalar parameter, are superseded with computer controlled knowledge-based systems which include abstract multi-parameter models of physical phenomena, with test signal generation to assist in the measurement process, and iterative procedures to setup the initial model, guide the experiment, validate the results, and refine the structure [1], [3], [5].

The paper will present generic architecture of a knowledge-based instrumentation system, where functions and structure no longer follow the same route, and consequently cannot be described by a simple and unique model. The system is directed to knowledge-based diagnosis in process engineering, as the primary application area. Particular attention will be paid to two substantial aspects of such systems: model building and knowledge representation and reasoning.

FUNCTIONAL MODEL

A real-time fault diagnostic system, applied to an engineering domain, has extensive advantages based on its expanded capabilities; by continuously accessing and analysing data from its changing knowledge base and environment it reasons on necessary actions to take in real time. In order to fulfil the expected functional requirements extensive problems must be solved. The system must provide gua-

ranteed response times, completing its reasoning and performing actions within a deterministic time dead-line. It must take into consideration the asynchronous time occurrence of events, and perform nonmonotonically.

The functional model of an intelligent real-time fault diagnostic system is presented in Figure 1. There are several distinctive modules interacting between the user and the process. In the explanation and analysis of these modules we must concentrate on its employed performance and interactions, rather than on the technological implementation and structure.

The system must support multiple levels of abstraction in functional modelling, and hierarchical decomposition of a process (plant) to assist in capturing the dynamics at the desired granularity. This is the purpose of the model builder with the attached domain knowledge. The plant functional representation consists of nodes and connections. Each node represent a functional entity (process) that produces and consumes material, energy, and information [2], [3]. Each node (process) is described by a number of subprocesses (immediate descendants of the node), states (steady state, transient state etc.), input-output parameters (transducers, setpoints, actuators), measurement and control algorithms, failure modes (variable limits, energy and material balance violation), alarms and failure modes association, and a fault propagation graph.

Model builder is a tool which enables construction of multiple levels of abstraction. To couple its components with the real process, one needs other functional modules, as demonstrated in Figure 1. Measurement and control module is presented in every manually and automatically driven process. Its purpose is to capture process variables, guide them according to some algorithms and record time histograms. This module is tightly coupled with data analysis and interpretation functions, which can be further backed with some expert knowledge [3]. User can access process parameters on various hierarchical levels and update the knowledge base.

Implementation design, with design knowledge module, bonds the actual measurement and control module with abstract structure, generated by the model builder. Its purpose is to keep the model within the definite bounds, informing the user on violation of various physical environment constraints. It also ensures that the system is always in a consistent and reliable state, advising the user on critical situations that are obviously not intended, e.g.: the user didn't connect objects in the process model.

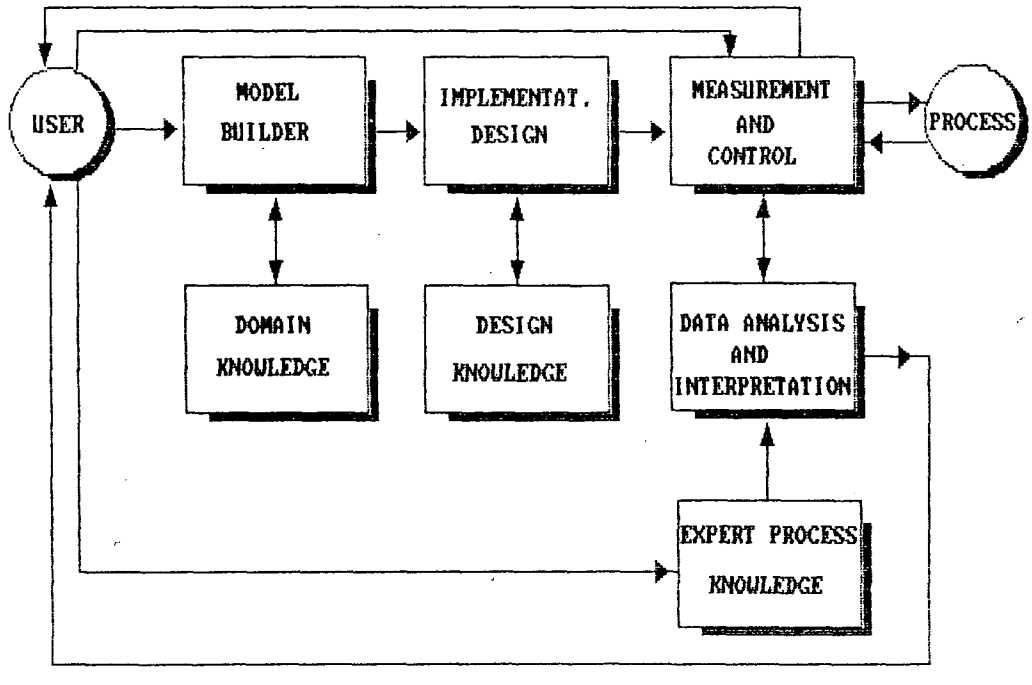


Figure 1. Functional model of an intelligent process instrumentation system.

STRUCTURAL MODEL

Intelligent instrumentation system structural model is based on hardware and software components which form the unified infrastructure. Basic building blocks, arranged as hierarchical layers, are presented in Figure 2.

The elements of the physical and system layers are those commonly found in all computer based instrumentation systems. The physical layer generally includes modular single processor, multi processor or distributed processor architecture, with main and auxiliary storage units, and input-output devices. It is very tightly coupled with specific interface devices, which act as data acquisition and control subsystem. Interface layer transform process primary data (such as temperature, pressure, level, et.) into a domain suitable for further processing. Data acquisition is a high priority task governed by the process real-time constraints, and occurring asynchronously with respect to the rest of the instrumentation system.

System layer implements an extended or virtual machine. Permitting a top-down view, it provides the user with a convenient interface to the physical layer. An alternative, bottom-up view may describe it as a manager of resources (processors, memories, etc.). A trend in modern operating systems is to move code up into higher layer, leaving a minimal kernel. The usual approach is to implement most of functions in application processes, where a process (client) sends a request to a server process, which does the work and sends back the answer. The kernel handles communication between clients and servers. By splitting the operating system into parts (various services), each part is small and manageable. The client-server model is also suitable to be used in distributed systems, since client need not know where the server is actually performing (locally or remotely).

Next layer, as presented in Figure 2, consists of several tools, or modules, which form a conceptual bridge between low level machine instructions and high level notions, embedded in the application layer. It is a set of modules (library) that implements diverse functionalities in the information processing domain.

Procedural (imperative) language compilers (C, Pascal) are the most traditional elements of that layer. Because of the expanding scope and complexity of problems found in intelligent instrumentation systems, some new languages must be included, i.e. declarative (functional) problem oriented languages such as LISP and PROLOG. Declarative languages have referential transparency, i.e. variables have definite value or unknown value, rather than the modified value (by assignment operation) in procedural languages. Stabilising variables allow finite mathematical techniques to be applied to code fragments, a step towards better code verification. Programmers, without assignments, can declare what effects should

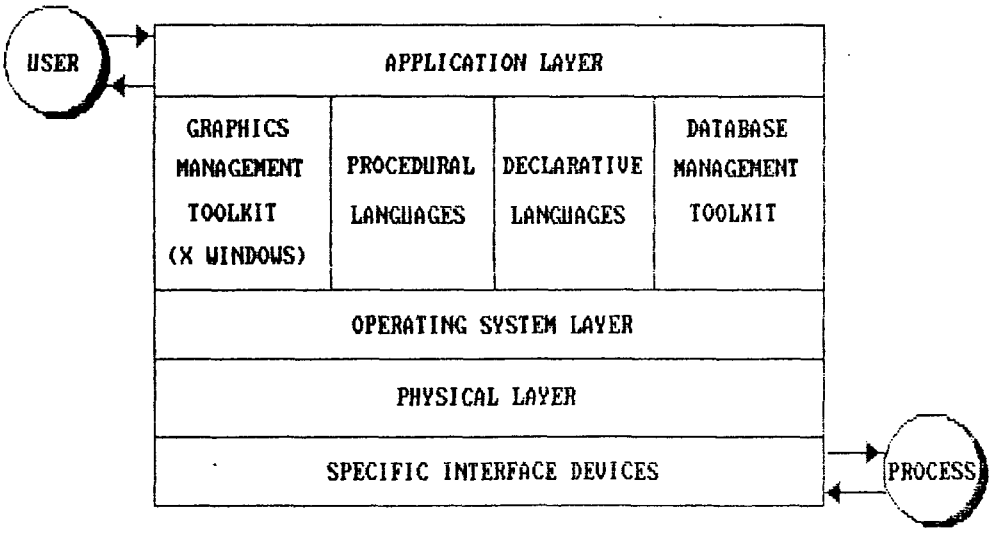


Figure 2. Structural model of an intelligent process instrumentation system.

produce what outcomes, rather than prescribe the sequences of processing. Removing the explicit sequencing, allows for the possibility of parallel processing, since a function can be executed whenever all its inputs exist.

Graphics management toolkit is essential in developing visual user interface during model building, and other phases of system operation. The industry standard X Window system [4], is based on already mentioned client-server paradigm, and supports one or more screens containing overlapping windows or subwindows. A single X server can provide display services for any number of screens. Within a screen there are windows arranged in strict hierarchies. At the top of each hierarchy is a root window. An application program can create an arbitrary deep tree of each screen. X library of functions provides graphics, text, and raster operations for windows. X programming environment has adopted object oriented approach. The fundamental abstraction and data type is the widget, a combination of an X window and its associated semantics, and which is dynamically allocated and contains state information. Every widget belongs to exactly one widget class that is statically allocated and initialised, and that contains the operations allowable on widgets of that class. Logically, a widget class is the procedures and data that is associated with all widgets belonging to that class. These procedures and data can be inherited by subclasses.

Real-time instrumentation systems must capture full process dynamics by periodically observing its state space. This is accomplished by data acquisition task, which is triggered by a real-time clock (asynchronously to other system activities) updating the blackboard type global database. Hence a database management toolkit is an essential element in the integrated system, as presented in Figure 2. The most needed functions are creation of a database, modifying its structure, adding, deleting, viewing and editing individual records and groups of records. There must be mutual exclusion guard mechanisms to ensure database integrity when reading and writing actions are concurrent, when these activities are not synchronised. Knowledge representation, for intelligent instrumentation application, can also take a database form [5]. Both knowledge base and database are data structures in which information is stored and from which information is retrieved. While semantic interpretation adopt a knowledge base perspective, the implementation mechanisms view it as a database. An item in a database is not merely a value but a piece of procedural and declarative knowledge, hence a database query mechanisms are of principal concern.

MODEL BUILDING

Complex structure of a contemporary process plants require model builders and model interpreters based on the graphical toolkit, that enable process engineers to develop and store models in a pictorial form. Such an environment enables selection and linking visual objects or icons together as a block diagram to create complete solution. The graphs are the result of an abstraction, and as such are at the same abstract level as mathematical functions. For manipulation and storage these objects need a formal representation. Several approaches to a formalisation are presented in [6], with particular attention paid to NP-completeness, i.e. intractable or computationally hard problems. The concrete situation is less serious though, due to the bounds of terminal screens, and restrictions to planar graphs.

We will restrict to a class of interactive systems in which the state of the object is visualised within the display and there is a consistent relationship between what is manipulated (state) and what is perceived (display). The other design issue is that physical characteristics of the display limit to part of the state only. Hence a strategy is required to make the whole state readily accessible through the display. In order to make distinction between state and display, we need two algebras (models): a state algebra (S, C) consisting of state and commands on the state, and a display algebra (D, O) consisting of display and operators that manipulate display. C is a set of state transformations ($S \rightarrow S_1$), and similarly O is the set of display transformations ($D \rightarrow D_1$). The display is a view of some aspects of state, so there is a mapping ($v: S \rightarrow D$) that produces a visual representation of some or all of the state. The mapping from state to display affirms that the display is updated as the state is transformed by commands C. The model builder that confirms to that basic principles, and enable developers to represent relationship among system components and the tasks to be performed on these components in a natural way,

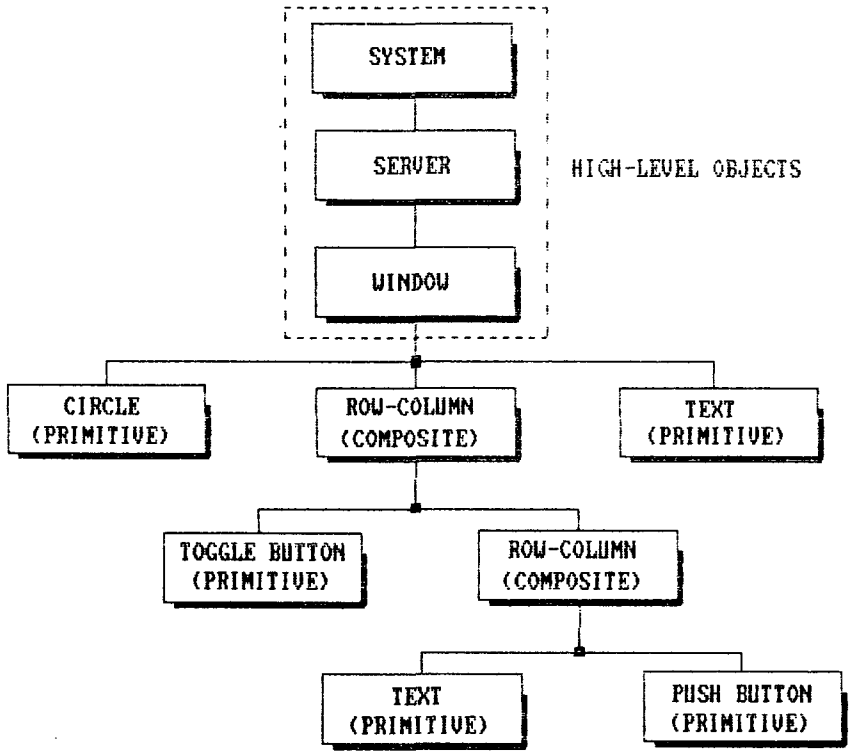


Figure 3. Simple hierarchy of graphic objects.

is the one that takes object oriented approach [7], [8].

All operations in such a system are done with objects, resulting in a hierarchy of objects. Building this hierarchy starts with creating window objects and then placing graphics and widget objects into the windows. To activate the object in the window some of the attributes of the object can be changed (e.g. foreground or background colour). When an application uses objects to display data values, it can make calls to functions to update the data values. Every object belong to a group or hierarchical level. A simple object hierarchy, consisting of four groups (high-level, composite, primitive, and low-level objects) is presented in Figure 3.

The key features of object-based systems are the concepts of polymorphism, messaging, and inheritance. Polymorphism allow different objects to share a common operational interface. When an operation is invoked, the function dynamically determines the object type and executes the appropriate code. Objects communicate with each other through massaging. Sending a message to an object requests that objects to perform some action, usually the manipulation of its internal data. Inheritance provides the ability to create incremental definitions of objects. The new definition extends the existing definition by adding data to the object representation, by adding new methods (procedures), and by extending the definition of existing methods.

If, as an example of multiple inheritance, one wishes to combine window object and a scroll bar object to produce a scrollable window, the code skeleton (in C++) might look like:

```
class Window
{ public:
    Window(int top, int left,
           int bottom, int right);
    ~Window (); }
class SrollBar
{ public:
    ScrollBar(int top, int left,
              int bottom, int right);
    ~ScrollBar(); }
class ScrollWin : Window, ScrollBar
{ public:
    ScrollWin(int top, int left,
              int bottom, int right);
    ~ScrollWin(); }
```

Based on the presented object-oriented framework, model builder consists of a powerful set of functions for creating and activating user interface components. Models of physical objects can be effectively created to provide context-specific information that the end user can react to very quickly, which is an essential feature in process control application domain.

KNOWLEDGE REPRESENTATION AND REASONING

The predominant issue in designing the intelligent instrumentation systems is embedding the domain knowledge, design knowledge, and specific expert knowledge, as indicated in Figure 1. The problems of integration real-time instrumentation systems with knowledge based systems arise due to the significantly different data and control structures in these two types of systems. Moreover, there is a tradeoff in knowledge representation techniques, i.e. between expressiveness of a representational language and its computational tractability.

There are two major properties that the structures in a knowledge based system must satisfy. First, it must be possible to interpret hypothesis as propositions representing the overall knowledge of the system, and second, the symbolic structures within a knowledge based system must play a causal role in the behaviour of that system. The role of a knowledge representation subsystem is to manage a knowledge base and present a picture of the world based on that representation. Within this role several approaches are possible, and each produces distinctive tradeoff between expressiveness and tractability.

The most prevalent type of knowledge is using the first-order logic (FOL), i.e. facts about the world. The end result of this process is a first-order knowledge base: a collection of sentences in FOL. The question as to whether or not the truth of a particular sentence is implicit in the knowledge base, reduces to whether or not a sentence is a theorem of FOL. The question-answering operation becomes one of theorem proving, which has pseudo-solutions if we relax our notion of correctness. Strictly speaking it is computationally intractable [9].

Restricting a knowledge base to a database form, characterises exactly the positive instances of the various predicates, and the range of uncertainty is limited [5]. It is shown that knowledge representation through database, taking closed world assumption, is easier to use due to the fact that inferences reduces to calculations. Because of this expressive limitation, knowledge representation is computationally more tractable. The comparative analysis of representational forms in a database and explicit knowledge base shows, that a comprehensive expert system shell can be devised above data structures within the database itself.

The second restrictions is representing a knowledge base in a logic program form (e.g. PROLOG). As in the database case, it must also contain the closed world assumption, but this time an inference process (an execution of a program) may be required to answer questions. This representational form is much more manageable then the general case since the inference can be split into a pattern matching facts retrieval component, and a search component, partially under user control, that uses nonatomic sentences.

Knowledge representation through semantic network only contains unary and binary predicates. This form can be represented by a labelled directed graph. Hence, a certain kind of inference can be performed by simple graph-searching techniques. Any mechanism that speed up graph-searching can be used directly to improve performance of inference in a knowledge base of this form.

Finally, the frame description form of the knowledge representation, is an elaboration of the semantic network one, with the emphasis on the structure of types (frames) in terms of their attributes (slots). One doesn't have to state explicitly the hierarchy of types. Rather, the descriptions themselves implicitly define the taxonomy. Analytic relationship, like subsumptions and disjointness, are properties of the structured types that are not available in a semantic network, where all of the types are atomic.

The presented formalisms take different positions on the tradeoff between expressiveness and tractability. There is no apparent winner for an application in the domain of process monitoring, control, and fault diagnosis, and an analysis of the logical form is needed in each particular case [3], [9], [10].

CONCLUSION

Intelligent instrumentation systems are based on integrating artificial intelligence techniques with conventional software technologies. An architecture that support symbolic and numeric computations is thus needed. Functional model and structural model of such a system reveal the computational heterogeneity. Signal processing modules, databases, reasoning agents, etc., have to be implemented and connected to form an integrated system. This heterogeneity also requires a programming environment that supports multiple programming paradigms.

Focusing on the intelligent instrumentation systems applied to process monitoring and diagnosis, the decomposition of the system under analysis into a hierarchical model of varying resolution and backtracking along fault propagation paths, are natural ways of modelling and reasoning about failures. The model building has settled around the techniques that employ object-oriented and windowing methodology (X windows). Models of physical objects can be created and interrelated to provide context-specific information. Knowledge representation and reasoning on faults in a complex process system, still follow very diverse approaches. Even though there is a strong relationship between graphical modelling of causal faults propagation (fault trees, signed and unsigned digraphs, event digraphs etc.) and semantic networks, some research indicate, that rule-based logic program models agree quite well with human performance.

REFERENCES

[1] J.Sztipanovits, Design of Intelligent Instrumentation, Proc. of the 1st Conf. on Artificial Intelligence Applications, IEEE Computer Society, Dec. 1984, pp. 490-495.

[2] S. Padalkar et all., Real-Time Fault Diagnosis, IEEE Expert, Vol. 6, (1991), No.3, pp.75-85.

[3] N.Bogunovic, Coupling an Expert System with Industrial Real-Time Processes, Proc. of the 16th Annual Conference of IEEE Industrial Electronics Society, IECON'90, IEEE Catalogue No. 90CH2841-5, Vol.II, pp.1281-1286.

[4] T. O'Reilly et all., X Window System User's Guide, O'Reilly and Associates, Newton, MA, 1987.

[5] N.Bogunovic, Knowledge Representation in a Database Environment for Intelligent Instrumentation Applications, Proc. of the IASTED International Conference, Artificial Intelligence and Neural Networks, July 1-3, 1991, Zurich, Acta Press, Zurich, pp.141-144.

[6] P.Gorny, M.J.Tauber (Eds.), Visualisation in Human-Computer Interaction, Lecture Notes in Computer Science (439), Springer-Verlag, Berlin, 1990.

[7] J.Sztipanovits et all., Programming Model for Distributed Intelligent Systems, Proc. of the NASA Conf. on Artificial Intelligence for Space Applications, Huntsville, Ala., USA, Nov. 13-14, 1987, pp. 365-373.

[8] K.Roger at all., An Overview of the HP Interactive Visual Interface, Hewlett-Packard Journal, Vol.41 (1990), No.5, pp.6-10.

[9] R.J.Brachman, H.J.Levesque (Eds.), Readings in Knowledge Representation, Morgan Kaufmann Publishers, Inc., 1985, pp. 42-70.

[10] N.H.Narayanan, N.Viswanadham, A Methodology for Knowledge Acquisition and Reasoning in Failure Analysis of Systems, IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-17, No.2, March/April 1987, pp. 274-288.

SLOGI UPORABNIŠKIH VMESNIKOV

INFORMATICA 1/92

Keywords: human-computer interaction, user-interface, user-interface style, direct manipulation, user-interface design

Matjaž Debevec, Dali Đonlagić, Martin Leš
Tehniška fakulteta Maribor
Elektrotehnika, računalništvo in informatika

POVZETEK

Komunikacija človek-računalnik postaja vse pomembnejša na vedno več področjih. Pomemben del te komunikacije je uporabniški vmesnik, ki omogoča prenos informacij med človekom in računalnikom. Uspešnost uporabniško prijaznega programa je odvisna predvsem od preprostosti in hitrosti njegove uporabe. Z razvojem računalništva so nastajali različni slogi uporabniških vmesnikov, v začetku alfanumeričnih, nato grafičnih.

Uspešnost uporabniško prijaznih programov je v veliki meri odvisna od pravilne izbire elementov iz različnih slogov uporabniških vmesnikov. V članku so podani in opisani najpogostejši slogi uporabniških vmesnikov. Opisane so njihove lastnosti in ideje, njihove prednosti in slabosti in povezave med različnimi slogi.

USER-INTERFACE STYLES

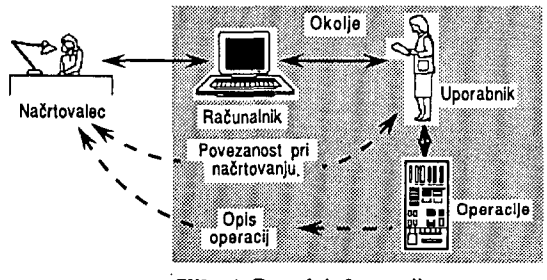
Human-Computer communications is becoming important in many fields. Important part of this communication is user interface, which enables the information transmission between man and the computer. The efficiency of the user friendly program depends on simplicity and speed of its use. With the development of software ingeneering various user interface styles, first the alpha-numerical then the graphical one, have appeared.

The efficiency of user-friendly programmes dependends on elements, correctly chosen from various user interface styles. In the paper the most frequently used user interface styles are given and described. Their characteristics and ideas, their advantages and weaknesses and connections between various styles are described.

1. Uvod

Začetki komunikacije človek-računalnik segajo v 60 leta, ko so poskušali razširiti uporabo računalnikov na druga področja. S tem se je močno razširil krog uporabnikov. Leta 1963 so začeli razvijati prve sisteme z dodeljevanjem časa, od katerih je najbolj znan MIT-MAC. Od takrat se je počasi povečal pomen ostalih tipov uporabnikov, ki so želeli uporabljati računalnike. Temu ustrezno so se razvijali različni slogi uporabniških vmesnikov. Najopaznejši premik v razvoju uporabniških vmesnikov je nastal z uvajanjem grafičnih delovnih postaj v 80 letih[1].

Uporabniški vmesnik je splošno gledano informacijski kanal, ki prenaša informacije med človekom in računalnikom. Vendar pri tem ne gre samo za celoten potek od razvoja programa do njegove uporabe in po potrebi do ponovnega načrtovanja za izpopolnitev programa. Slika 1 kaže realen **pretok informacij** med načrtovalcem, računalnikom in uporabnikom. Načrtovalec načrtuje program tako, da prilagodi njegovo delovanje z uporabnikovim delom in zahtevami. Pri tem mora načrtovalec upoštevati tudi različne tipe uporabnikov. Nujna je začetna povezava načrtovalca z uporabnikom, saj ni nujno, da načrtovalec pozna vse zakonitosti uporabnikovega dela. Tudi po daljšem delu s programom se lahko zgodi, da pride do sprememb, ki jih mora načrtovalec pozneje vstaviti v program. Vmesnik mora biti zato čim bolj odprt, kar pomeni, da bi lahko uporabljali enak vmesnik, na primer, za vodenje krmilnih naprav (industrijski procesi) ali za vodenje simuliranih naprav (simulator motorja). Zaradi tega ni enostavno izbrati ustreznega sloga uporabniškega vmesnika. V večini primerov izberemo, glede na okolje, v katerem bo uporabnik delal, elemente iz različnih slogov uporabniških vmesnikov. Postavlja se vprašanje, kateri slogi so najbolj primerni za določeno okolje. Zato mora načrtovalec dobro poznati njihove lastnosti, prednosti in slabosti. V nadaljevanju so opisani različni slogi uporabniških vmesnikov in njihove značilnosti[2].



Slika 1: Pretok informacij

2. Slogi uporabniških vmesnikov

V zadnjem času nastopajo naslednji slogi uporabniških vmesnikov[3]:

- **WYSIWYG** ("What you see is what you get")- vidiš to, kar si podal,
- **direktna manipulacija,**
- **ikone,**
- **multimedialni dialog.**

Skupna, glavna značilnost zgornjih uporabniških vmesnikov je potek komunikacije v grafičnem načinu in je zato pri delu potrebna zmogljiva grafična postaja.

V tem poglavju opisujemo lastnosti in ideje posameznih slogov, njihove prednosti, slabosti in povezave med različnimi slogi. Na koncu je tudi kratek opis drugih slogov uporabniškega vmesnika, ki ne spadajo direktno pod grafične uporabniške vmesnike. Ti so:

- **izbira po menuju,**
- **ukazna vrstica,**
- **dialog v naravnem jeziku,**
- **dialog s vprašanji in odgovori.**

Za uporabniške vmesnike lahko seveda uporabimo kombinacijo različnih slogov, vendar mora biti kombinacija pravilno izbrana in primerna glede na tip uporabnikov.

2.1. WYSIWYG slog

WYSIWYG (izgovori se "wiz-ee-wig") je zelo pomemben slog grafične predstavitve informacij. Predstavitev sistema, procesa, operacij, ki jih uporabnik vidi na zaslonu odgovarjajo realnemu, naravnemu izgledu. Večina dobrih programov, ki vsebujejo grafično predstavitev, vključujejo tudi komponente WYSIWYG.

Dobri urejevalniki besedil, na primer, uporabljajo WYSIWYG slog: Besedilo, ki ga želimo poudarjeno izpisati na tiskalnik, se tudi poudarjeno izpiše na zaslonu. Z urejevalnikom besedila, ki ne vsebuje WYSIWYG sloga, mora uporabnik uporabljati krmilne ukaze za besedilo.

Za primer si izpišimo naslednji stavek:

> V tem stavku vidimo <B>poudarjen<N>, <I>poševen<N> in <U>podčrtan<N> tekst.

ki definira naslednji izpis na tiskalnik:

> V tem stavku vidimo **poudarjen,** *poševen* in <u>podčrtan</u> tekst.

V tem primeru vidimo, da se moramo najprej naučiti pomena krmilnih ukazov in si ustvariti abstrakcijo izgleda na tiskalniku. Dokler uporabnik besedila ne izpiše, nima predstave o tem, ali je vse krmilne ukaze pravilno vpisal. Pri zelo dolgem besedilu je v takšnih primerih možnost napake zelo velika. Še največ krmilnih ukazov je potrebno pri matematičnih izrazih, kjer lahko nastopa nepregledna množica krmilnih kod. WYSIWYG slog v teh primerih bistveno izboljša pregled nad vnosom.

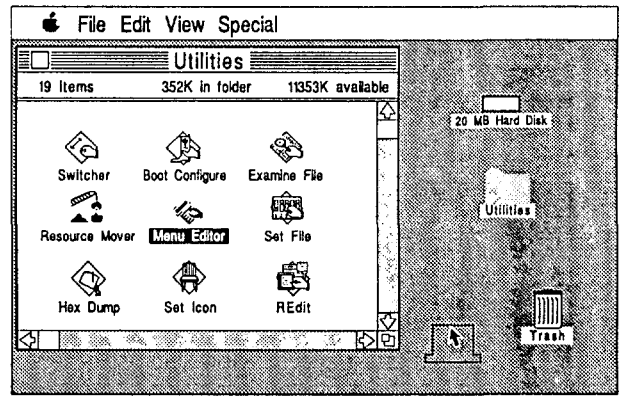
WYSIWYG slog ima tudi nekatere slabosti. Kjerkoli nastopajo zmogljive grafične postaje z veliko resolucijo in z mnogimi barvami, lahko nastopi problem, kako vse te informacije prenesti na papir, še posebej, če ni na voljo enako zmogljivega tiskalnika.

Lahko se pojavi tudi nasprotna situacija. Uporabnik ima na voljo zelo slab uporabniški vmesnik z WYSIWYG slogom, s katerim ne more točno vedeti, kaj se bo izpisalo na tiskalniku. Na voljo ima, na primer, v urejevalniku besedila, neko končno množico fontov, ki se prikažejo na zaslonu. Na tiskalniku pa ima možnost izpisa večje množice fontov, kot jih ima na zaslonu. Velikost in tip fonta na zaslonu zato včasih ne sovpada s fontom na tiskalniku.

2.2. Direktna manipulacija

Uporabniški vmesnik direktne manipulacije je sestavljen iz **objektov,** njihovih lastnosti in iz medsebojnih povezav, nad katerimi izvajamo **operacije**. Te operacije so tudi vizuelno predstavljene in jih izvajamo s pomočjo vizuelnih interaktivnih tehnik, kot na primer s puščico na zaslonu, ki predstavlja pomik s pomočjo miške ali grafične table. Ukazov ne izvajamo v smislu izbire po menuju ali s tipkanjem ukazov, ampak jih izvajamo kot akcije vizuelne predstavitve. Ta predstavitev je lahko besedilo, kot na primer ime objekta ali lastnosti in **grafični simbol**, ki ga imenujemo tudi **ikona**. V nadaljevanju je podrobneje opisan pomen grafičnih simbolov, ikon v vizuelni predstavitvi.

Primer uporabniškega vmesnika direktne manipulacije je predstavljen na sliki 2. V zgornjem desnem kotu je ikona za trdi disk, pod njim je ikona direktorija, ki je v sivi barvi, kar pomeni, da je odprta. Na levi strani zaslona je prikaz odprtega direktorija s poimenovanimi ikonami, ki predstavljajo datoteke v tem direktoriju. Datoteka, od katere vidimo le obrise okoli puščice, se pomika proti ikoni za smeti, ki je v spodnjem desnem kotu.



Slika 2: Zaslon računalnika Macintosh

Disketne enote in datoteke so torej v tem slogu predstavljene kot ikone. Pomik ene ali več datotek iz ikone, ki predstavlja prvo disketno enoto v ikono, ki predstavlja drugo disketno enoto, je enak pomenu kopiranja teh datotek iz ene v drugo disketno enoto. Če, na primer, datoteko pomaknemo v ikono za smeti (Trash), se datoteka zbriše.

Izraz direktna manipulacija je prvi skoval Shneiderman[4], ki je predstavil tudi mnogo različnih primerov, ki kažejo tipične značilnosti direktne manipulacije. Te so med drugim vnaprejšnja predstavitev besedila, oblike, kot je pozneje izpisan na tiskalniku (WYSIWYG), pri izpisu tabel so potrebne oblike, ki kratko in jasno predstavljajo podane tabele in drugo. Video igre, CAD/CAM sistemi, velike podatkovne baze, procesni sistemi so prav tako primeri programov, kjer najdemo direktno manipulacijo.

Shneiderman je predstavil tudi prednosti direktnih manipulacijskih tehnik:

- začetniki se hitreje naučijo osnovnih ukazov,
- izkušen uporabnik se lažje takoj posveti svojemu problemu,
- število sporočil o napakah se bistveno zmanjša,
- večja prilagodljivost različnim tipom uporabnikov.

Uporabniški vmesnik direktne manipulacije je sestavljen iz različnih komponent . Na najvišjem nivoju direktnega manipulacijskega sistema je grafična predstavitev **metafore**, kot na primer površina **pisalne mize**, ki predstavlja liste in orodja. Druge metafore so lahko predstavitev **modela procesnega sistema, elektronskega laboratorija, merilnega sistema** in drugih. Na tem najvišjem nivoju lahko uporabnik uporablja objekte, ki so prikazani v čim realnejši obliki v oknih na zaslonu. Oken je lahko na zaslonu poljubno mnogo in se lahko med seboj pokrivajo, lahko se prikličejo in uporabljajo, tako kot bi ravnali s kupi listov na pisalni mizi.

Za manipulacijo objektov in vodenje uporabljamo različne načine, kot so:

- **upravljanje,**
- **vodenje,**
- **menuji,**
- **dialogna področja.**

Upravljanje pomeni direktno vodenje objekta, kot na primer merilnega instrumenta v programu za merjenje. To vodenje se lahko po izbiri ustreznega objekta z miško direktno opazuje na zaslonu. Izvršujemo lahko množico operacij, manipulacij, kot na primer, spremembo velikosti, razteg, rotacijo in drugo.

Vodenje pomeni izbira funkcij ali vnos različnih parametrov vhoda. Na voljo imamo različne oblike vodenja:
- **tipke** (navadne tipke, radio tipke, kontrolne tipke),
- **okviri** (izpisni, vnosni),
- **valuatorji** (skale, drsni rob (Scroll bars), prirejeni valuatorji, primerni za določen program).

Menuje prištevamo med najelementarnejše dele vodenja, ki jih uporabljamo tudi v drugih slogih. Menuji lahko nastopajo tudi sami zase. V nekaterih uporabniških okoljih so menuji samo zbirka tipk, ki jih aktiviramo s pritiskom na miškino tipko. Najbolj pogosto uporabljene oblike menujev so **pull-down, pop-up** in **kaskadni menu.**

Dialogna področja uporabljamo za izpis sporočil za preverjanje ali potrditev pomembnih ukazov, kot na primer "Ali naj shranim datoteko pred izhodom iz programa?"). Ta področja uporabimo tudi za vnos podatkov ali parametrov. Značilna lastnost teh področij je, da uporabnik najprej odgovori na vprašanje ali vstavi podatek, ki ga sporočilo zahteva, preden lahko dela dalje s programom. Ta tehnika je manj direktno manipulativna kot ostale, ker nudi nižji nivo komunikacije s človekom.

Ostali elementi direktne manipulacije so povezani s predstavitvijo informacije, kot na primer **ikone**, s katerimi lahko manipuliramo in izvajamo določene operacije.

Tipične **operacije** po izbiri objekta s pomočjo vhodne, izbirne naprave so:

- premik objekta na drugo lokacijo,
- brisanje objekta,
- kopiranje,
- rezanje in dodajanje odrezanega objekta,
- odpiranje in zapiranje datotek.

Direktno manipulacijo velikokrat predstavljajo kot najboljši slog uporabniškega vmesnika. Ta slog je v resnici izredno zmogljiv in zelo enostaven za učenje. Vendar lahko postane za izkušenega uporabnika počasen. Pri tiskanju datoteke s poljubnim imenom, na primer, mora uporabnik pri direktni manipulaciji najprej odpreti okno za izbiro datotek, poiskati datoteko in jo določiti z vhodno napravo (običajno z miško). Nato mora poiskati ikono ali izbrati ukaz za tiskanje. Če pa uporabnik že pozna ime datoteke in ve v katerem direktoriju se nahaja, je zanj mnogo hitreje, če vtipka ukaz: "Tiskaj datoteke". To je še posebej primerno pri velikem številu datotek, kjer bi se moral uporabnik pri iskanju datoteke v oknu direktorija pomikati po straneh. To je samo eden od mnogih primerov, pri katerih je vnos v ukazno vrstico lahko mnogo hitrejši kot direktna manipulacija. Omejitve se kažejo tudi v tem, da potrebuje direktna manipulacija zelo zmogljivo grafično postajo, skupaj z miško. Poleg tega ni enostavna realizacija slike, objektov in težjih ukazov na zaslonu, ki predstavljajo metaforo uporabnikovega delovnega okolja.

Priporočljivo je, da kombiniramo direktno manipulacijo z ukazno vrstico. V nekaterih primerih se celo zgodi, da nekaterih ukazov ne moremo vključiti v direktno manipulacijo. Zato mora uporabnik imeti možnost vpisa teh ukazov v ukazno vrstico. To pomeni, da se pri načrtovanju uporabniškega vmesnika ne opredelimo samo za en slog.

2.3. Uporabniški vmesnik z ikonami

Ikona je **slikovna predstavitev** objekta, akcije, lastnosti in drugih konceptov. Načrtovalec uporabniškega vmesnika ima za predstavitev določenega koncepta na voljo uporabo ikone ali besed. V tem primeru ikone ne smemo povezovati z direktno manipulacijo. Besedilo sicer lahko direktno manipuliramo enako dobro kot ikone, vendar ikone zmanjšajo potreben prostor za izpis.

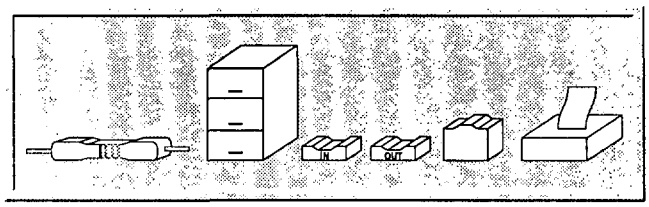
Vprašanje je, kaj je bolje uporabiti, ikone ali besedilo. Kot velika večina odgovorov, je tudi tukaj odgovor odvisen od same naloge in ciljev uporabnikovega dela. Ikone imajo veliko prednosti pred izpisom besedila. Dobro načrtovana ikona lahko pove mnogo več kot besede, zavzame mnogo manj prostora in je s tem razbiranje mnogo lažje. Če je ikona pravilno izbrana, lahko postane tudi jezikovno neodvisna, kar omogoča uporabo uporabniškega vmesnika v različnih državah.

Zato je potrebno pri načrtovanju upoštevati tri cilje:

- **razpoznavanje** - pomeni hitrost razpoznave pomena ikone,
- **pomnenje** - pomeni, kako dobro si lahko uporabnik zapomni pomen ikone,
- **razlikovanje** - pomeni, kako dobro se ikone med seboj ločijo po izgledu in pomenu.

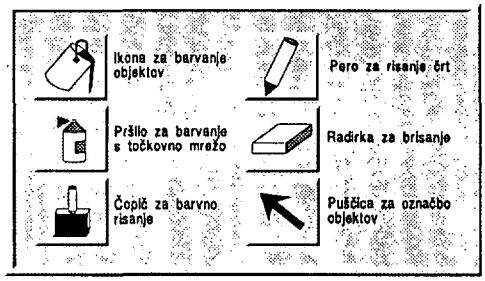
Ikone, ki predstavljajo objekte, se lahko relativno enostavno načrtujejo in jih imenujemo **objektne ikone.** Slika 3 kaže zbirko nekaterih ikon, ki predstavljajo objekte. Lastnosti objekta lahko enostavno predstavimo, če je vsaka vrednost predstavljena v pravilni grafični obliki. To so lahko na primer debelina črt, velikost znakov, električnih simbolov, razred uporov (v barvah) in drugo.

Ukaze nad objekti lahko prav tako predstavljamo z ikonami. Pri tem razlikujemo različne načrtovalne principe za izvedbo **ukaznih ikon.**



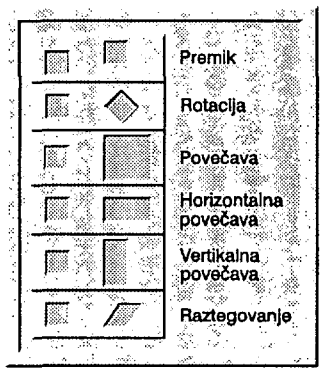
Slika 3: Objektne ikone

Prvi princip je, da z ikono **predstavimo objekt**, ki se pojavi tudi v realnem svetu, na primer, škarje uporabimo za rezanje, vedro barve za polnitev področij, okvirjev, svinčnik za risanje črt in drugo. Slika 4 kaže zbirko takšnih ukaznih ikon. Te ikone se uporabnik težje nauči, ker je potrebna najprej razpoznava, kaj ikona pomeni in kaj lahko z njo naredi. Zaradi dvokoračnega razumskega procesa je razumevanje in učenje daljše, kot pri enokoračnem, kjer je potrebno samo razpoznavanje pomena ikone.



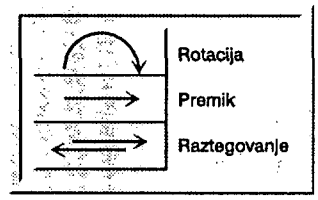
Slika 4: Ukazne ikone prikazujejo objekte za izvajanje določene naloge

Naslednji princip načrtovanja ukaznih ikon je **prikaz ukazov pred in za efektom**, kot je prikazano na sliki 5. Ta princip se uspešno uporablja tam, kjer so predstavitve objektov z ikonami enake, kot nastopajo v programu. Če nastopajo različni tipi objektov, ki s predstavitvijo nimajo nobene zveze, potem lahko uporabnik dobi občutek, kot da nad objekti, s katerimi trenutno razpolaga, ne more izvajati operacij, predstavljenih z ikonami. Zato se te vrste ikon pojavljajo le tam, kjer so tipi objektov enaki ali vsaj podobni objektom, prikazanih v ikonah.



Slika 5: Ukazne ikone, ki predstavljajo geometrične pretvorbe

Naslednji princip je **abstraktna predstavitev ukaza**. Tipičen primer je prikazan na sliki 6. Predstavitev je odvisna od različnih kultur in znanja v okolju, kjer se pojavljajo. Znak X lahko, na primer, nekje pomeni tudi brisanje.



Slika 6: Abstraktna predstavitev ukaznih ikon

Za uporabo ikon lahko rečemo, da je zelo primerna za grafične uporabniške vmesnike, kjer nudijo predstavitev množice objektov na zelo majhnem prostoru. Če so ikone skrbno izdelane, je lahko primerjanje in odčitavanje pomena ikon lažje, kot branje množice ukazov.

2.4 Multimedialni dialog

Interaktivne videosisteme, ki so povezani z ostalimi mediji, kot so zvočni vhod/izhod imenujemo **multimedialne dialogne sisteme**. Razvoj teh sistemov imajo nekateri za revolucionarnega[6]. Pri multimedialnih sistemih komunicira uporabnik z računalnikom s pomočjo video slik ali filma, ki mu predstavljajo čim bolj realno sliko delovnega okolja, in s pomočjo zvočnih signalov. Človek lahko, na primer, pri opazovanju procesa na zaslonu, predstavljenega z video sliko, poseže v proces z izgovarjanjem ukazov ali s pomikom kazalca na ustrezno mesto v procesu. To mesto lahko opazuje z dodatno kamero, kar mu poveča preglednost in ustrezno reagiranje. Na koncu lahko celoten uporabnikov postopek še enkrat preverjamo.

S pomočjo digitalnega shranjevanja in obdelave video slike ali zaporedja teh slik in shranjevanja zvočnih signalov se zato realni model (metafora) najbolj približa sliki realnega sveta.

Predpogoj za uporabo takšnih sistemov so zelo zmogljivi računalniki in uporaba optičnih pomnilnikov. Poleg tega so potrebni še zelo hitri postopki za kompresijo in dekompresijo zaporedja video slik v realnem času. V CD-ROM (Compact disk read only memories) pomnilnikih s kapaciteto približno 650 Mbytov se sicer lahko shrani 300.000 strani teksta, vendar samo 30 s dolg film, sestavljen iz video slik. S kompresijo slikovnih podatkov lahko dosežemo s temi CD-ROM-i kapaciteto za približno 72 min filma.

Drugi elementi, ki so potrebni za multimedialne dialoge, so **orodja** za avtorje dialoga, s katerimi načrtujejo in vodijo video in slišne podatke. Poleg tega so potrebni postopki za obdelavo slik, s katerimi lahko iz posnetih video slik izdelamo druge želene poglede na slike. Na MIT (Massachusetts Institut of Technology) razvijajo poleg teh orodij tudi različne primere, kot so interaktivna dokumentacija (razvoj mest), učno okolje (tuji jeziki), komunikacije (raziskovalne kooperacije) in drugo [7].

3. Ostali slogi uporabniških vmesnikov

Slogi, ki smo jih dosedaj opisali, pripadajo slogom, izvedenim za **grafične uporabniške vmesnike**. Ostali slogi sicer niso specifično namenjeni grafičnim uporabniškim vmesnikom, vendar jih kljub temu lahko uporabljamo v grafičnih uporabniških vmesnikih. Ti slogi so:

- **menuji,**
- **ukazna vrstica,**
- **naravni-pogovorni dialog,**
- **dialog s vprašanji in odgovori.**

3.1. Menuji

Menuji so najbolj razširjeni slog tako v grafičnih kot v negrafičnih programih. Osnovna prednost menujev je v tem, da uporabnik pri delu z njimi lahko uporablja samo kratkotrajni spomin, ker ima na voljo slike (tekstovni ali ikonski ukazi), ki mu pomagajo pri razpoznavi. To je ravno nasprotno pomnenju posebnih ukazov, ki si jih mora uporabnik vtisniti v dolgotrajni spomin. Zato so menuji še posebej primerni za začetnike, ki ne potrebujejo velikega naprezanja pri učenju dela s programom. Po drugi strani menuji nudijo tudi zmanjševanje potrebnih ukazov, česar drugi slogi niso zmožni. Z zmanjševanjem potrebnih ukazov mislimo na izbiro samo nekaj možnih ukazov iz celotnega nabora ukazov. Možni ukazi so predstavljeni s črno barvo na svetlem ozadju ali z belo na temnem ozadju, ostali, ki jih ne moremo aktivirati, pa s sivo.

3.2. Ukazna vrstica

Uporaba ukazne vrstice je najstarejši način komunikacije z računalnikom. Ta slog lahko uvrstimo tudi med dialoge z iniciativo uporabnika, med katere prištevamo tudi menu. Uporabnik pri uporabi ukazne vrstice najprej vpiše besedo, kodo ali kakšen drug ukaz. Pri tem računalnik takoj izvede akcijo, ki ustreza vnosu. Program vsebuje svoj lastni programski jezik, ki vsebuje veliko množico ukazov in ki jo lahko poljubno razširjamo. Tega jezika se mora uporabnik temeljito naučiti. Vrstica, kamor uporabnik vpisuje ukaze, običajno ne vsebuje nikakršnih dodatnih informacij. V nekaterih primerih pa lahko vsebuje **pozivni stavek** (prompt), ki ga poda uporabniku. Nato uporabnik glede na sporočilo ustrezno odgovori. Ta slog dialoga ima malo različnih oblik. Glavna razlika je v tipu informacije, ki jo uporabnik vpiše in se interpretira v računalniku. Ti tipi so lahko:

- besede (predstavljajo imena funkcij, ki se naj izvedejo),
- črke, okrajšave,
- logični in matematični izrazi,
- akcijske kode (kratke kombinacije različnih ukazov).

Programi z ukazno vrstico ne vsebujejo mnogo zahtev, kaj naj uporabnik naredi. Z enostavnim vtipkanjem ukazov dobi izkušen, uporabnik občutek prostosti in neomejenosti, saj nudijo takšni uporabniški vmesniki visoko stopnjo prilagodljivosti. To pomeni, da lahko ukaze med seboj poljubno križamo in pri tem niso potrebna točno določena zaporedja ukazov.

Prednosti te vrste programov so:
- enostavna razširitev ukazov,
- splošna uporabnost vseh funkcij v vsakem trenutku,
- pri rutinskem delu skrajša čas reševanja problemov.

Slabosti pa so:
- čas učenja za delo s programom je mnogo daljši, še posebej pri veliki množici ukazov,
- večja možnost napak,
- samo redno, intenzivno delo je učinkovito,
- primeren je samo za izkušene uporabnike, ki znajo hitro tipkati.

3.3. Naravni, pogovorni dialog

Naravni, pogovorni dialog je najpomembnejši medij komunikacije med ljudmi. Ta dialog je pogosto prikazan kot alternativa za interaktivne sisteme. Če bi računalnik lahko razumel naše izgovorjene ukaze, bi ga lahko znal uporabljati praktično vsak uporabnik. Najprimernejši bi bil med drugim za slepe ljudi. Vendar je danes izdelava vmesnika v naravnem jeziku še vedno drzno dejanje. Današnji razpoznavalniki govora z velikim številom besed se morajo posebej prilagajati uporabnikovemu glasu. Če tega posebnega treninga ni, se lahko zelo pogosto pojavijo napake. Največja težava teh razpoznavalnikov je še vedno razpoznava hitrega, tekočega govora zaradi nenatančnih premorov med posameznimi besedami. Naravni, pogovorni dialogi so primerni za:

- dialogne sisteme s vprašanji in odgovori,
- poizvedovanje v podatkovnih bazah,
- kombinirani govorni in grafični sistemi.

Zelo primeren je naravni dialog v kombinaciji z direktno manipulacijo. V naravnem dialogu se velikokrat pojavlja dialog, pri katerem pokažemo na neko stvar in povemo, kaj je ali povemo, kaj bi radi z njo. Prav tako lahko z miško izberemo želeno ikono in izgovorimo ukaz, ki manipulira z njo. Z raziskavami so prišli do ugotovitve, da na ta način lahko uporabniki delajo do 60% hitreje kot pri klasični direktni manipulaciji.

3.4. Dialog s vprašanji in odgovori

Dialog s vprašanji in odgovori prav tako kot menuji spada med dialoge z iniciativo računalnika in je izmed vseh slogov najbolj preprost. Program postavi vprašanja, na katere uporabnik odgovori z enim od možnih odgovorov. V najpreprostejših oblikah so ti odgovori lahko samo "DA" ali "NE". Dialog je zelo primeren za **binarne izbire**. Primer takšnega procesa najdemo v elektroniki in medicini za reševanje problemov binarnih operacij in postavljanje diagnoz. Na podlagi odgovorov program išče po različnih poteh, dokler ne pride do končnega rezultata ali diagnoze. Prav tako to tehniko srečamo pri vpisovanju formularjev ali različnih podatkov, kot so datumi, letnica rojstva, številka stanovanja in drugo, kjer so prostori za vpis oblikovani tako, da uporabnik točno ve, kaj mora vpisati v posamezen prostor.

Tabela 1 nam kaže primerjavo med najbolj znanimi dialognimi slogi glede na čas učenja, hitrost uporabe, pojavnost napak, razširljivost in ali je potrebno znanje tipkanja.

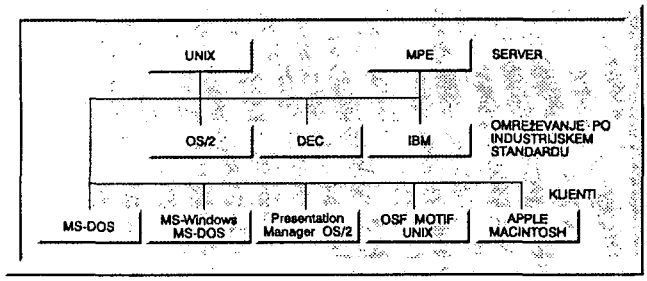
4. Zaključek

V zadnjem času se torej na različnih računalniških sistemih vse bolj uveljavljajo **grafični uporabniški vmesniki**. Podobno, kot je potekal razvoj televizije, poteka sedaj tudi razvoj računalniških zaslonov. Na začetku so bili na voljo samo črno-beli računalniški zasloni z zelo majhno resolucijo in brez grafike. V prihodnosti bo razvoj, tako kot pri televiziji, popolnoma izpodrinil črno-bele zaslone in se bodo uveljavili visoko resolucijski zasloni z zmogljivo grafiko. Vsi večji računalniški sistemi, ki so danes uveljavljeni, razvijajo grafične uporabniške vmesnike. DEC uporablja DEC-Windows, IBM-DOS ima Windows, OS/2 uporablja Presentation Manager in UNIX Motif ali Open Look. V prihodnosti se bodo morali vsi ti različni uporabniški vmesniki na nek način med seboj povezati, da se bodo lahko podatki uporabljali tako v enem kot v drugem uporabniškem vmesniku. To velja še posebej, kadar različne sisteme želimo povezati med seboj v mreže. Ker v večini uporabniških vmesnikov manipuliramo z objekti, bo potrebno natančno določiti mrežne objekte in manipulacije nad njimi v mreži. Slika 7 kaže primer povezave različnih uporabniških sistemov v mrežo.

V bližnji prihodnosti bi moral računalnik imeti naslednje lastnosti:

- biti čim bolj "oseben",
- nuditi bo moral uporabo kakršnegakoli programa, ne glede na tip računalniškega sistema,
- uporabljati bo moral mrežo,
- vključiti elektronsko pošto, glasovno pošto in FAX,
- dostop do različnih podatkovnih baz po celem svetu.

Prvi koraki v tej smeri so narejeni s pomočjo grafičnih uporabniških vmesnikov in s poskusi po standardizaciji njihovih slogov. Novejši grafični uporabniški vmesniki vsebujejo objekte, s katerimi enostavno manipuliramo. Prvi večji napredek je tudi uvedba **OLE** (Object Linking and Embedding), ki bo v prihodnosti vse bolj pogost. Tu gre predvsem za to, da lahko podatke, besedila, slike, ki smo jih naredili v poljubnem programu enostavno prenesemo v kakšen drug program. Poleg tega to tudi pomeni, da vse pomembnejše nastavitve, kot so uporaba perifernih enot ali različnih tipov črk, enako uporabljamo ne glede na program. Na ta način se bistveno poenostavi delo z računalnikom in poveča ustvarjalni učinek človeka ob računalniku.



Slika 7: Prikaz mrežne povezave z različnimi računalniki

LITERATURA

1 Fähnrich, K.P., "Software-Ergonomie", (State of the Art; 5), München; Wien: Oldenbourg, 1987

2 Thimbleby H., "User Interface Design", ACM Press, Addison-Wesley Publishing Company, New York, 1990

3 Foley J.D., A. van Dam, S.K. Feiner, J.F. Hughes, "Computer Graphics - Principles and practice", Addison Wesley Publishing Company, New York, 1990

4 Shneiderman B., "Direct Manipulation: A Step beyond Programming Languages". IEEE Computer, 1983, 16, str.57-69

5 Ziegler J.E., "Direct Manipulation Techniques for the Human- Computer Interface", Eurographics '90 Technical Report Series, Tutorial Note 11.

6 Fox,E.A, "The coming revolution of interactive digital video", Communications of the ACM, 32(7),1989,794-801

7 Mackay,W.E., Davenport G.,"Virtual Video Editing in Interactive Multimedia Applications", Communications of the ACM, 32(7),1989, 802-810.

	WYSI WYG	Direktna manipulacija	Menuji	Vpis v okvirje	Ukazna vrstica	Naravni dialog	Dialog s vprašanji-odgovori	Multimedialni dialog
Čas učenja	kratek	kratek	srednji	kratek	dolg	kratek	kratek	kratek
Hitrost *uporabe*		srednja	srednja	velika	velika	srednja	majhna	majhna
Pojavnost napak	majhna	majhna	majhna	majhna	velika	velika	majhna	majhna
Razširljivost	majhna	majhna	majhna	srednja	velika	velika	velika	velika
Znanje tipkanja		ni potrebno	ni potrebno	potrebno	potrebno	ni potrebno	potrebno	ni potrebno

Tabela 5.1: Primerjava med osmimi različnimi slogi uporabniških vmesnikov

PREDIKCIJA ČASOVNIH SERIJ Z NEVRONSKIMI MREŽAMI

INFORMATICA 1/92

Keywords: Feedforward Net, Recurrent Net, Deterministic Chaos

Tjaša Meško, Andrej Dobnikar
Fakulteta za elektrotehniko
in računalništvo, Ljubljana

V prispevku je analizirana sposobnost nevronskih mrež pri napovedovanju determinističnega kaosa, ki je primer nelinearnega dinamičnega sistema. Opisana je primerjava med večnivojsko nevronsko mrežo s povratno povezavo in brez povratne povezave. Obema nevronskima mrežama med fazo učenja določimo število vozlov v skritem nivoju glede na hitrost konvergence. Izkazalo se je, da pri predikciji determinističnega kaosa navadna FF (feed forward) nevronska mreža ne zaostaja bistveno za rekurentno (recurrent), pri hitrosti učenja jo celo prekaša.

NEURAL NETWORKS FOR TIME SERIES PREDICTION. This contribution describes the results of testing a feedforward and a recurrent neural network on determinintic chaos prediction. Both networks are trained by the back-propagation algorith which varies the number of hidden units depending on the convergence speed. It is shown that the feedforward network converges equally well as the recurrent one, even with a substantialy lower number of hidden units.

1 Uvod

Pri obravnavanju naravnih pojavov ali pri iskanju kakšnih drugih zakonitosti se pogosto pojavi vprašanje napovedovanja prihodnosti. Kot primer bi lahko navedli vremensko napoved ali pa napovedovanje dogajanja na borzi. V primeru, da poznavanje sistema lahko zapišemo z rešljivo enačbo, lahko napovedujemo prihodnost, brž ko so določeni začetni pogoji. Če take enačbe ne poznamo, pa poiščemo empirične zakonitosti sistema in glede na le-te poskušamo bolj ali manj natančno napovedovati.

Pri slednji metodi se soočamo s problemi, kot je na primer šum. Periodičnost sistema je lahko zakrita s šumom v tolikšni meri, da se obnašanje sistema zdi popolnoma naključno.

Večnivojske nevronske mreže se v zadnjem času pogosto uporabljajo pri predikciji prihodnjih vrednosti časovnih serij s pomočjo učenja na preteklih primerih. Če se osredotočimo na časovne serije s šumom, je nevarnost, da se bo mreža primere in s tem tudi šum preveč naučila (*overfitting*). Pri tem je ključnega pomena velikost nevronske mreže (število vozlov v skritem nivoju).

2 Večnivojske nevronske mreže

Analiza in procesiranje časovnih serij sta mogoči na več načinov. V prispevku bomo obravnavali pristop z nevronskimi mrežami, kjer bomo uporabili dve znani topologiji: FF (*feed-forward*) in rekurentno (*recurrent*) topologijo. Naš cilj je primerjalno ovrednotiti obe topologiji predvsem z vidika hitrosti konvergence in velikosti topologije.

Obravnavali bomo le mreže z enim skritem nivojem. Mrežo učimo z back-propagation metodo [Rumelhart et al.,86]. Napaka mreže je definirana kot

$$E = \sum_{n} \sum_{i=1}^{m} (o_i^* - o_i)^2 \qquad (1)$$

kjer je m število vozlov izhodnega nivoja, o_i je izhod i-tega izhodnega vozla, o^* je i-ta komponenta učilnega vektorja, n pa število elementov baze.

FF mreža z enim skritim nivojem je predstavljena na sliki 1. Izhod vozla i skritega (izhodnega) nivoja je definiran kot

$$o_i = f(\sum_{j} w_{ij} y_j + w_{i0})$$

kjer je w_{ij} utež povezave med i-tim skritim (izhodnim) vozlom in j-tim vhodnim (skritim) vozlom, y_j je izhod vhodnega (skritega) vozla j in f je sigmoidna funkcija.

Primer rekurentne mreže oz. mreže s povratno povezavo dobimo, če FF mrežo z enim skritim nivojem dopolnimo s kontekstnim nivojem [Elman,90], ki je prav tako skriti nivo (glej sliko 2). Povezave od skritega do kontekstnega nivoja imajo konstantno utež 1, uteži ostalih povezav pa se med učenjem spreminjajo v skladu z back-propagation algoritmom. Izhode skritega nivoja si mreža torej v kontekstnem nivoju zapomni.

Pri učenju večnivojskih mrež se zmeraj pojavi problem optimalne konfiguracije mreže (število vozlov v skritem nivoju) pri dani velikosti dopustne napake (glej enačbo 1). V

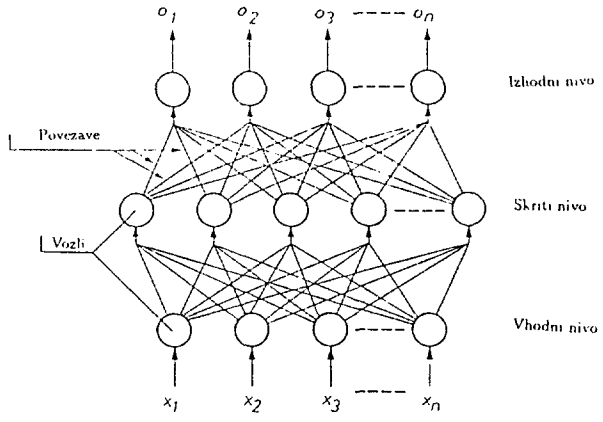


Figure 1: *Mreža brez povratne povezave (FF).*

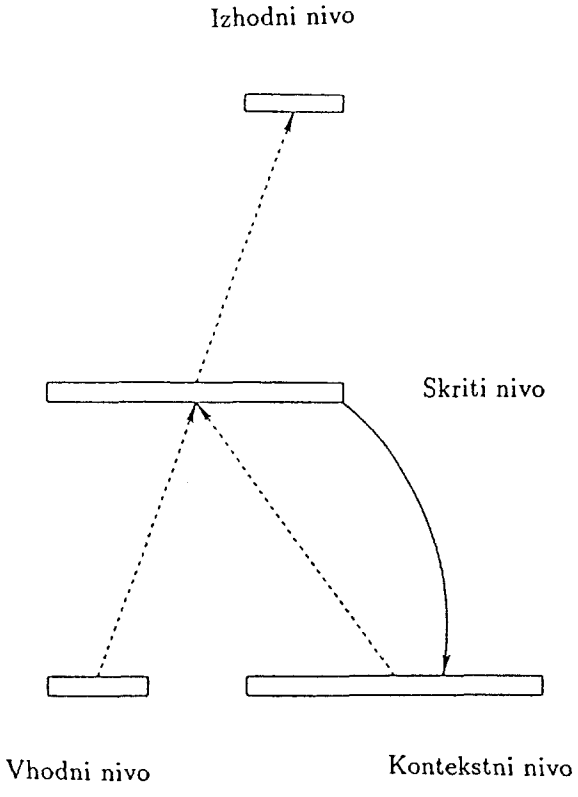


Figure 2: *Mreža s povratno povezavo med skritim in kontekstnim nivojem (rekurentna mreža).*

ta namen lahko pri učenju mrež z oz. brez povratne povezave uporabimo algoritem, ki spreminja število skritih vozlov [Hirose et al.,91] glede na hitrost padanja napake. Če napaka po nekem številu sprememb uteži ne pade za več kot 1%, dodamo vozel skritemu nivoju. Ko je napaka dovolj majhna, postopoma odvzemamo vozle v skritem nivoju, dokler napaka spet ne naraste preko dovoljene meje.

3 Deterministični kaos

Za deterministični kaos sta pri predikciji časovnih serij značilni dve lastnosti [Weigend et al.,90]:

- Ker nemoč napovedovanja narašča eksponentno s časom, je možnost dolgoročne predikcije izključena.
- Mogoča je kratkoročna predikcija: časovne serije, ki se zdijo naključne, so lahko bile generirane z deterističnim sistemom.

Kaotično naravo predstavimo z vrednostjo x_t, ki je funkcija predhodnih d vrednosti časovne serije

$$(x_{t-1}, x_{t-2}, \ldots, x_{t-d}) \mapsto x_t$$

Vektor $(x_{t-1}, x_{t-2}, \ldots, x_{t-d})$ leži v d-dimenzionalnem prostoru časovne zakasnitve (*lag space*).

Standarden primer determinističnega kaosa je iterativna kvadratna preslikava na interval enote

$$x_t = 4x_{t-1}(1 - x_{t-1})$$

Ta preprosta enačba ilustrira nekatere ključne probleme determinističnega kaosa.

- Navidezna naključnost. Kljub temu, da je časovna serija generirana s pomočjo popolnoma determinističnega sistema brez šuma, se zdi sekvenca naključna.
- Kratkoročna predvidljivost. Ne glede na to, da se zdi časovna serija naključna, jo je mogoče dokaj natančno napovedati že s pomočjo prilegajoče se funkcije skozi množico zadnjih dveh točk $\{x_{t-1}, x_t\}$ iz preteklosti.
- Dolgoročna nepredvidljivost. V vsaki iteraciji izgubimo en bit informacije [Weigend et al.,90].

Deterministični sistemi se razlikujejo od stohastičnih po ciljih predvidevanja. Pri determinističnih sistemih želimo napovedati vrednost člena časovne serije v prihodnosti, ki naj bo čim bliže dejanski vrednosti v tistem trenutku. Pri stohastičnih sistemih pa lahko ima neko stanje več mogočih prehodov v različna stanja, ki se po vrednostih parametrov lahko precej razlikujejo. Predvidevanja so v tem primeru odvisna od verjetnosti stanj. Deterministični sistemi s šumom lahko predstavljajo poseben primer stohastičnih sistemov.

Če hočemo narediti model napovedovanja determinističnega sistema (lahko je kaotičen ali pa vsebuje šum), je potrebno izvesti naslednje tri točke:

1. Izberemo prostor časovne zakasnitve (*lag space*).
2. Aproksimiramo predhodne časovne serije $\{x_t(x_{t-1}, x_{t-2}, \ldots, x_{t-d})\}$ z gladko površino. Različni pristopi pri napovedovanju časovnih serij se razlikujejo predvsem po izbiri osnovnih elementov (polinomi, sigmoidi, ipd.).
3. Izberemo funkcijo cene, ki pove, kako dobra je aproksimacija.

V naslednjem poglavju bomo kot model napovedovanja kaotičnega sistema uporabili večnivojsko mrežo dveh topologij: FF in rekurentno.

4 Rezultati testov

Mreža s povratno povezavo in mreža brez povratne povezave sta imeli po en vhodni

	Učna baza	Testna baza
FF	0.0188	0.0192
Rekurentna	0.0186	0.0190

Table 1: *Napaka na učni in testni bazi pri uporabi večnivojske mreže z in brez povratne povezave pri kriteriju konvergence 0.02.*

	Učna baza	Testna baza
FF	0.0097	0.0098
Rekurentna	0.0098	0.0520

Table 2: *Napaka na učni in testni bazi pri uporabi večnivojske mreže z in brez povratne povezave pri kriteriju konvergence 0.01.*

in en izhodni vozel. Število vozlov skritega nivoja smo določili z algoritmom, ki spreminja število skritih vozlov [Hirose et al.,91] glede na hitrost padanja napake. Pri rekurentni mreži je število skritega nivoja in konteksta zmeraj enako, zato ju povečamo ali zmanjšamo naenkrat. Pri testu obeh mrež je bil uporabljen deterministični kaos z enačbo

$$x_t = 3.8x_{t-1}(1 - x_{t-1})$$

pri čemer je $x_0 = 0.5$. Deterministični kaos smo izbrali zato, ker ga je relativno preprosto generirati.

Učna baza je bila sestavljena iz prvih stotih elementov časovne serije, testna baza pa iz nadaljnjih stotih elementov. Vsota kvadratov napak (glej enačbo 1), za vsak primer posebej na testni in učni bazi, je podana v tabeli 1 in 2. Razvidno je, da je razlika med napako na testni in na učni bazi relativno majhna, kar je posledica dejstva, da ni bilo šuma. Opazna razlika med napako na testni in učni bazi je le pri rekurentni mreži pri velikem številu iteracij.

Pri prvem poskusu je bila vrednost vsote kvadratov napak 0.02 pogoj za konvergenco na učni bazi. Primerjavo med nevronskima mrežama vidimo na sliki 3. Pri drugem poskusu je bil parameter konvergence

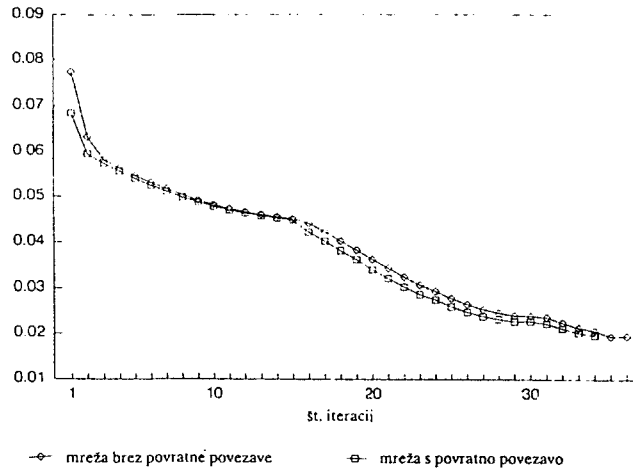


Figure 3: *Primerjava napake pri večnivojski mreži z in brez povratne povezave. Parameter konvergence ima vrednost 0.02.*

zmanjšan na 0.01. To je povzročilo precej daljše učenje in tudi povečalo število vozlov v skritem nivoju (glej sliko 4). Za dosego kriterija konvergence 0.02 sta pri obeh topologijah zadoščala dva vozla v skritem nivoju (pri rekurentni še dva vozla v kontekstu). FF mreža je zmanjšala napako pod 0.02 po 36 iteracijah, rekurentna mreža pa po 34 iteracijah. Za dosego napake, manjše od 0.01, je FF potrebovala 21 skritih vozlov in 84 iteracij, rekurentna pa kar 23 vozlov v skritem nivoju in 23 vozlov v kontekstu ter 87 iteracij.

5 Zaključek

Mreža brez povratne povezave kaže na primeru deterministicnega kaosa glede na obseg topologije in konvergenco ugodnejše lastnosti kot rekurentna mreža. Pri majhnem številu iteracij je napaka rekurentne mreže nekaj manjša od napake FF, vendar se ta prednost s številom iteracij izgubi. Učenje rekurentne mreže je tudi precej bolj zamudno zaradi kontekstnega nivoja. FF mreža dobro posplošuje z učne na testno bazo ne glede na število iteracij, rekurentna pa posplošuje pri

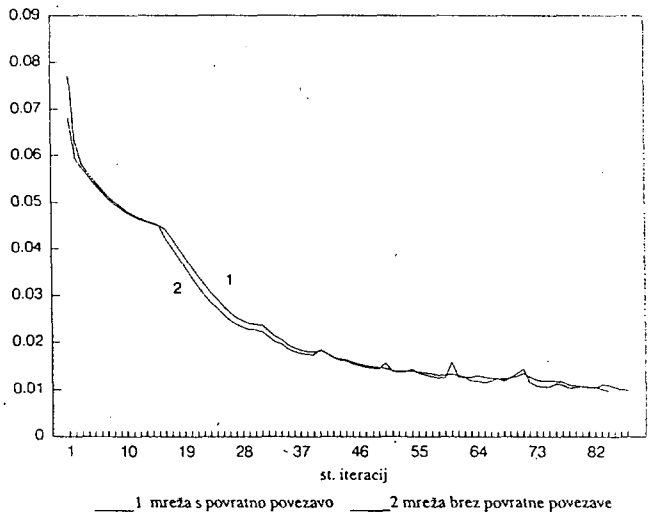


Figure 4: *Primerjava napake pri večnivojski mreži z in brez povratne povezave. Parameter konvergence ima vrednost 0.01.*

večjem številu iteracij slabše.

Literatura

[1] Elman,J.L. (1990). "Finding Structure in Time", Cognitive Science 14, 179-211, 1990.

[2] Hirose,Y., Yamashita,K. and Hijiya,S. (1991). "Back-Propagation Algorithm Which Varies the Number of Hidden Units", Neural Networks, Vol.4, pp.61-66, 1991.

[3] Rumelhart,D., Hinton,G. and Williams,R. (1986). "Parallel distributed processing: exploration in the microstructure of cognition", Vol. 1, MIT Press.

[4] Weigend,A.S., Huberman,B.A. and Rumelhart,D.E. (1990). "Predicting the Future: A Connectionist Approach", International Journal of Neural Systems, Vol.1, No.3, pp.193-209, 1990.

FDDI - LOKALNA MREŽA PRIHODNOSTI

INFORMATICA 1/92

Keywords: LAN, FDDI, Ethernet, MAN, backbone-networks

Marjan Bradeško, Ivan Pepeljak
Nil. d.o.o., Tesovnikova 88/a, Ljubljana

POVZETEK - Propustnost standardnih tipov lokalnih mrež kot sta Ethernet in Token ring postaja ozko grlo zlasti pri kompleksnih mrežah. FDDI - nova generacija lokalnih mrež s hitrostjo prenosa 100 Mbit/s preko optičnih kablov predstavlja kvalitetno osnovo za t.i 'backbone-mreže' kakor tudi za mestne mreže (MAN).

ABSTRACT - The throughput of the standard types of local area networks is becoming the bottleneck especially in complex networks. The FDDI - a new generation of local area networks with transmission speed of 100 Mbit/s over optical cables represents quality base for backbone-networks and metropolitan networks (MAN).

1. Ozka grla današnjih lokalnih mrež

Pri dosedanjih lokalnih računalniških mrežah (LAN - Local Area Network) prevladujeta dve topolgiji - vodilo in obroč. Najbolj razširjen predstavnik pri prvem tipu je Ethernet (IEEE 802.3), pri drugem pa Token ring (IEEE 802.5). Največja prenosna hitrost prvega je 10 Mbit/s, drugega pa 4 oziroma 16 Mbit/s. Dokaj zadovoljive hitrosti, dokler je mreža majhna oziroma komunikacijsko nezahtevna.

V večjih podjetjih in ustanovah obstaja več manjših lokalnih računalniških mrež (podmrež), ki so ponavadi povezane skupaj v večjo lokalno mrežo preko nekakšne hrbtenice (backbone). Pogosto je to Ethernet na debelem koaksialnem kablu (10 Base-5), katerega prenosna hitrost je še vedno omejena na 10 Mbit/s. Pri intenzivnem medmrežnem komuniciranju postane prav to razlog za ozko grlo celotne računalniške mreže določene ustanove. Lokalni promet v okviru posamezne mreže sicer omejimo z napravami kot je npr. bridge, toda intenzivnost medmrežnega komuniciranja nenehoma raste. Porazdeljenost podatkov kot tudi obdelav je vse večja, s tem pa tudi promet v računalniških mrežah.

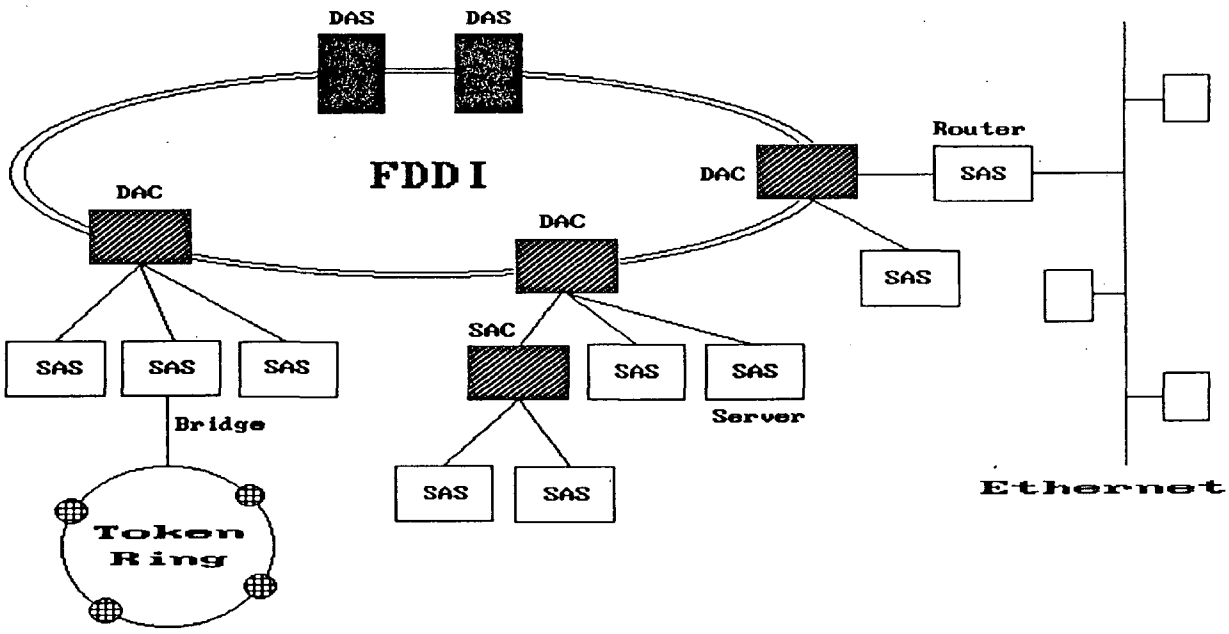
Kot prenosni medij v lokalnih računalniških mrežah se zaradi svojih dobrih lastnosti (npr. neobčutljivost na motnje iz okolice) pogosto pojavlja kabel iz optičnih vlaken. In prav na osnovi tega medija je v svetu nastal nov standard za lokalne računalniške mreže, skrit v kratico FDDI, ob kateri začne danes mnogim srce utripati hitreje.

2. FDDI - definicija in standardi

FDDI (Fiber Distributed Data Interface) je standard za novo generacijo lokalnih mrež. Standard na osnovi optičnega kabla omogoča prenos s hitrostjo 100 Mbit/s do 100 km daleč z največjo razdaljo 2 km (pri monomodnih kablih tudi več, glejte nadaljevanje) med dvema postajama in največ 500 postajami v mreži. Topološko gledano je FDDI dvojni nasprotnosmerni obroč z žetoni (dual counter rotating token ring), katerega osnovne komponente prikazuje slika 1. Velikost paketa v mreži je omejena na 4500 zlogov (Ethernet 1500 zlogov).

Standard FDDI (rev 6.2) je definiran s strani ameriške ustanove ANSI oziroma njenega komiteja X3T9.5. Razdeljen je v naslednje štiri dele:

- PMD (Physical Media Dependent) - definicija optičnih kablov, povezav, sprejemnikov in oddajnikov, optične specifikacije
- PHY (Physical Layer Dependent) - kodiranje /dekodiranje podatkov med PMD in MAC, definicija časovnih razmer in podatkovnih okvirov



Slika 1: Topologija FDDI in povezava z ostalimi mrežami

- MAC (Media Access Control) - definicija fizičnih izvornih/ciljnih naslovov, preverjanje in popravljanje napak, sprejem in generiranje podatkovnih okvirov
- SMT (Station Management) - definicija konfiguracij, spremljanje dogajanja v mreži, povezava z modelom OSI

3. Prednosti FDDI

Med številnimi faktorji, ki narekujejo uvajanje FDDI, so:

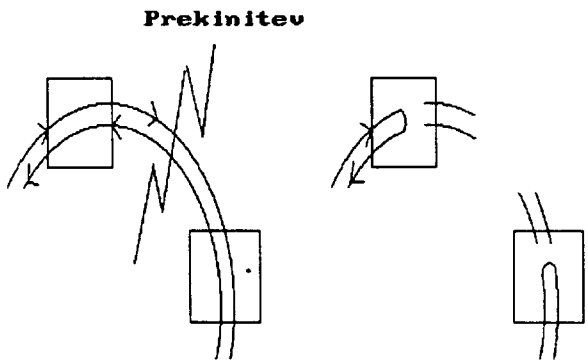
- integracija obstoječih lokalnih mrež v večjo backbone-mrežo
- vse večja porazdeljenost podatkov in obdelav
- vse večje procesorske zmogljivosti
- grafično intenzivne aplikacije
- aplikacije tipa odjemalec/strežnik (client/server)
- aplikacije na osnovi standarda X-Windows
- zahteve po povezavah na večje razdalje
- zahteve po veliki zanesljivosti delovanja

FDDI se zaradi svojih lastnosti uveljavlja tudi na širših območjih - uspešno pokriva tudi področje mestnih mrež (MAN - Metropolitan Area Network). Taka mreža denimo uspešno deluje v ameriškem Houstonu, kjer so uspešno rešili tudi težave (varnost), ki se pojavljajo ob uporabi relativno zaprte mreže v javnem okolju (več različnih ustanov) [3].

Ena najpomembnejših prednosti, ki jih ima FDDI pred dosedanjimi tipi mrež, je visoka prenosna hitrost, saj s 100 Mbiti na sekundo predstavlja idealno sredstvo za integracijo več posameznih podmrež v večjo backbone-mrežo. Obenem tako široka pasovna širina omogoča implementacijo zahtevnih aplikacij s porazdeljenimi bazami podatkov in procesiranje slik v realnem času. Omeniti pa moramo, da hitrosti FDDI v začetni fazi uvajanja ne bodo polno izkoriščene in bodo prišle do izraza šele s prihodom zmogljivejših procesnih enot, ki bodo prednosti hitrega prenosa znale izrabiti. Podobno je bilo ob uvajanju Etherneta - hitrosti prenosa so se gibale nekako okrog 1 Mbit/s. Prilagoditev je že narejena tudi pri višjenivojskih protokolih kot je npr. TCP/IP. Ti protokoli znajo izkoriščati novo velikost paketa v FDDI-mreži, ki je omejena na 4500 zlogov.

Druga prednost je velika zanesljivost delovanja, ki jo zagotavljajo vgrajeni mehanizmi odpravljanja napak, obenem pa že sam optični medij, ki je neobčutljiv na motnje iz okolice. FDDI ima dva nasprotnosmerna obroča, od katerih je eden primarni, drugi pa sekundarni. Po primarnem praviloma poteka ves promet, sekundarni pa je prost. V primeru okvare (prekinitve) primarnega obroča se aktivira sekundarni obroč. V

primeru okvare (prekinitve) obeh obročev na istem mestu se v obeh sosednjih postajah v bližini mesta okvare prevežeta (wrap) primarni in sekundarni obroč, ki spet tvorita zaključen krog (slika 2). S tem je obenem tudi lokalizirano mesto okvare, ki ga je v Ethernet-mrežah včasih težko najti.



Slika 2: Prevezava dvojnega v enojni obroč

Tretja prednost pri FDDI je manjša občutljivost mreže na dodajanja novih postaj vanjo. Pri Ethernetu včasih to povzroči precejšen padec zmogljivosti mreže. FDDI ima lahko naenkrat v mreži več žetonov (token), s tem pa tudi več paketov. Posamezni postaji ni potrebno čakati, da pride njen paket z žetonom nazaj (okoli po obroču), saj bi to pri tolikšnih razdaljah lahko trajalo predolgo. Da bo mreža čimbolje izrabljena, lahko postaje generirajo nove žetone pred vrnitvijo poslanega paketa. Časovni mehanizmi kroženja žetonov omogočajo deterministično določitev časa dostave posameznega paketa (kar je zlasti pomembno pri prenosu govora in gibljive slike).

Celotna dolžina mreže, ki jo omogoča FDDI, je kar 100 km (za razliko od 1.37 km pri Ethernetu), kar je dodatna velika prednost, saj so s tem uspešno pokrite tudi zahteve mestnih mrež (MAN). LAN-tehnologija s tem prodira na področja, ki so bila do nedavnega pokrita s tehnologijo WAN (Wide Area Networks).

4. Komponente FDDI

Na sam fizični medij, torej na dvojni obroč pri FDDI, so priključene naslednje vrste naprav (prikazane na sliki 1):

- postaje z enojno povezavo (SAS - single attachment stations)
- postaje z dvojno povezavo (DAS - dual attachment stations)
- koncentratorji (concentrators), ki imajo prav tako lahko enojno ali dvojno povezavo (SAC, DAC)

SAS - postaje z enojno povezavo se priključujejo na FDDI z enim samim dvosmernim optičnim kablom (priključek S). Priključno mesto je lahko katerikoli koncentrator oziroma vozlišče (hub). Priključevanje preko koncentratorjev ščiti sam dvojni FDDI-obroč pred okvarami posameznih postaj. SAS-postaje, ki se na koncentratorje priključujejo v konfiguraciji drevesa, predstavljajo najcenejši način povezovanja postaj v FDDI.

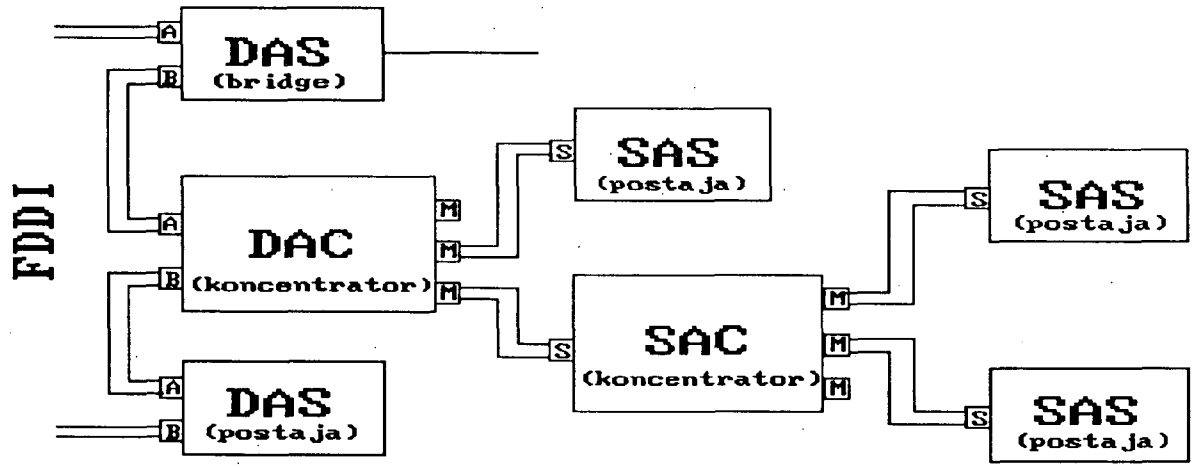
DAS - postaje z dvojno povezavo so priključene neposredno na sam dvojni (primarni, sekundarni) FDDI-obroč. Vsaka postaja DAS ima dva dvojna konektorja (priključka A in B). V primeru okvare pride v DAS-postajah, ki mejita na okvarjeno mesto, do prevezave (wrap) primarnega in sekundarnega obroča v enojni obroč. V DAS-postajah obstaja tudi stikalo, s katerim je možno preprečiti prevezavo obročev v primeru, da postaja ni vključena (ni pa okvarjena). Na ta način oba obroča optično obideta postajo (optical bypass). Vendar je v takem primeru treba upoštevati razdalje med delujočima postajama - slabljenje svetlobe je tolikšno, da razdalje dveh kilometrov, določene v standardih, ni zaželeno presegati.

Koncentratorji predstavljajo povezovalno mesto na dvojni FDDI-obroč za SAS- in DAS-postaje kakor tudi za druge koncentratorje. Vsebujejo več priključnih vrat (priključek M) za naštete postaje. Koncentrator lahko igra vlogo vozlišča (hub) za določeno delovno skupino (podmrežo). Uporaba koncentratorjev za SAS-postaje ščiti dvojni FDDI-obroč pred okvarami posameznih SAS-postaj. Koncentratorji so lahko priključeni neposredno na dvojni FDDI-obroč - označujemo jih z DAC (Dual Attachment Concentrator). Zaradi večje fleksibilnosti imamo tudi koncentratorje z enojno povezavo - SAC (Single Attachment Concentrator). Vloga koncentratorjev in postaj je razvidna iz slike 3.

5. Topologije FDDI

Četudi smo v uvodu rekli, da je FDDI topološko gledano dvojni nasprotnosmerni obroč, dopušča standard ANSI X3T9.5 še nekaj možnih različic.

Najenostavnejša (in najšibkejša) možnost je uporaba samostojnega koncentratorja s priključenimi postajami. Druga možnost je uporaba dvojnega FDDI-obroča z DAS-postajami, priključenimi neposredno nanj. Najbogatejša pa je konfiguracija, kakršna je prikazana na sliki 1. Tu imamo dvojni FDDI-obroč, nanj pa priključene tako DAS-postaje kakor tudi DAC-koncentratorje. Le-ti tvorijo poddrevesa vseh



Slika 3: Povezave FDDI-naprav

vrst postaj kakor tudi večjih lokalnih mrež. V primeru povezav lokalnih mrež v FDDI-mrežo nastopa SAS- oziroma DAS-postaja v vlogi bridgea oziroma routerja. Prav ta poddrevesa, povezana na dvojni FDDI-obroč preko koncentratorjev, omogočajo relativno enostaven nadzor in upravljanje mreže (network management) kot tudi spreminjanje konfiguracij.

Koncentratorji ne samo, da zagotavljajo veliko lokalno fleksibilnost brez vplivov na samo konfiguracijo dvojnega FDDI-obroča, pač pa povečujejo tudi zanesljivost same FDDI-mreže. V primeru neposrednega priključevanja postaj na FDDI-obroč bi lahko zaradi več hkratnih napak prišlo do segmentacije cele mreže - dobili bi več nepovezanih segmentov. Priključevanje preko koncentratorjev pa ima to prednost, da okvare posameznih postaj nimajo nobenega vpliva na sam osnovni FDDI-obroč.

6. Integracija obstoječih lokalnih mrež v FDDI

Za povezavo lokalnih mrež kot sta Ethernet in Token ring z FDDI-mrežo so potrebni bridgei oziroma routerji (uveljavljenega slovenskega izraza ni). Oboji so na voljo v obliki SAS- oziroma DAS-postaj.

Bridge deluje na drugem sloju komunikacijskega OSI-modela. Omogoča povezavo več fizičnih mrež v enotno logično mrežo. Bridge filtrira nepotreben promet na mreži, današnji FDDI-bridgei pa omogočajo tudi fragmentacijo paketov. S tem je namreč preprečena izguba daljših paketov. FDDI namreč na drugem sloju modela OSI dovoljuje do 4500 zlogov dolge pakete, Ethernet pa le 1500.

Pri bridgeih je treba opozoriti še na vrsto protokola. Nekateri namreč uporabljajo transparentni oziroma transalcijski protokol, drugi pa t.i. enkapsualcijski (encapsulated). Slednji lahko komunicirajo le z napravami, ki poznajo isti protokol, kar pomeni, da moramo v takem primeru imeti le naprave istega proizvajalca. Pri bridgeih z translacijskim protokolom teh težav ni.

Routerji delujejo na tretjem, mrežnem sloju modela OSI in usmerjajo pakete glede na mrežne naslove. Današnji FDDI-routerji prav tako kot bridgei omogočajo fragmetacijo paketov in s tem povezavo mrež z različnimi dolžinami paketov. Routerji so lahko enoprotokolni ali večprotokolni. So počasnejši od bridgeov, zlasti večprotokolni, ki so tudi bistveno dražji.

7. Nadzor in upravljanje v FDDI-mreži

Del standarda ANSI X3T9.5 je.tudi standard za nizkonivojski nadzor in upravljanje (management) mreže FDDI, ki edini še ni v končni obliki. Imenuje se SMT (Station Management) in zagotavlja nazdor in upravljanje mreže na drugem sloju modela OSI. S tem še nista zagotovljena celoten nadzor in upravljanje mreže FDDI, zato so potrebni še dodatni monitorji in protokolni analizatorji. Zlasti slednji so v času postavljanja mreže in šele nastajajočega standarda SMT

neprecenljivega pomena za načrtovalce, vzdrževalce in administratorje mreže FDDI. Mrežni monitorji pa omogočajo spremljanje dogajanja na mreži in izdelavo statistik, ki pozornega vzdrževalca mreže lahko pravočasno opozorijo na porajajoče se težave v njej.

Trenutni, najbolj razširjeni standard za nadzor in upravljanje mrež, je SNMP (Simple Network Management Protocol). V bazo informacij za upravljanje mreže (MIB - Management Information Base) so že vgradili mehanizme, preko katerih je možen dostop do števcev in kontrolnih funkcij, definiranih v standardu SMT. S tem je možna integracija funkcij SMT v okolje aplikacij na osnovi SNMP, kar mnogi proizvajalci opreme za FDDI že tudi ponujajo.

8. Implementacija FDDI

Pri implementaciji FDDI-mreže sta bistvenega pomena sam medij - optični kabel in mrežni krmilnik - kartica za priključitev postaj. Mrežni krmilniki so trenutno še precej dragi (5000 - 10000 dolarjev), zato še ni možno razmišljati o priključitvi poljubne postaje v mrežo, vsaj ne v takem smislu kot pri obstoječem Ethernetu. Prav zato se kot najpogostejša oblika implementacije pojavlja FDDI kot hrbtenica (backbone) za integracijo več lokalnih računalniških mrež v posamezni ustanovi.

Prav posebna pozornost pa je potrebna pri samem mediju - optičnem kablu. Od ustreznega izbora in polaganja je odvisen ves nadaljnji razvoj FDDI-mreže. Standard za FDDI priporoča 62.5-mikronski multimodni (multi mode) kabel, ki zagotavlja največjo razdaljo med postajami 1-2 km. Standard pa predvideva uporabo tudi drugačnih kablov, npr. 50-mikronskega, kjer je največja razdalja med postajami manjša. Za 125-mikronske optične kable pa velja, da je razdalja med dvema postajam lahko daljša od standardiziranih 2 km.

Omeniti moramo še monomodni optični kabel (mono, single mode), ki omogoča večje razdalje med postajami - tudi do 40 km in tudi večje hitrosti prenosa (ki v FDDI niso specificirane). Monomodni kabli zaradi zahtevnejše tehnologije predstavljajo precej večji strošek pri postavljanju mreže FDDI. Pri monomodnih kablih je izvor svetlobe laser namesto LED-diod, ki jih srečujemo pri multimodnih optičnih kablih. Uporaba monomodnih optičnih kablov ni priporočljiva tudi v primerih, če se uporabniške postaje priključujejo neposredno na sam FDDI-obroč, saj bi iztaknjen konektor zaradi močnega laserskega žarka lahko povzročil nesrečo. Tudi zaključevanje, spajanje in povezovanje monomodnih kablov je zahtevnejše. Možno pa je z uporabo ustreznih repeaterjev tudi povezovanje in kombiniranje mono- in multimodnih optičnih kablov.

9. Prihodnost FDDI-mrež

Vsekakor pomeni FDDI novo generacijo lokalnih računalniških mrež na povsem novi kvalitetni ravni. S podeseterjeno propustnostjo (hitrostjo prenosa) glede na današnje LAN-e in mnogo večjimi razdaljami, tako med postajami kot v celoti, sega FDDI tudi izven področja lokalnih mrež. Vse bolj aktualen postaja na področju t.i. mestnih mrež (MAN), kjer enakovredno tekmuje z DQDB (Distributed Queue Dual Bus), ki je standardiziran v priporočilih IEEE 802.6.

V pripravi je tudi standard, ki bo kot medij za FDDI definiral tudi oklopljeno in neoklopljeno parico (shielded, unshielded twisted-pair). Nekateri proizvajalci opreme poleg tega že ponujajo rešitve tudi s tankim koaksialnim kablom (thin coax). Prilagoditev FDDI na omenjena dva medija bo pomenila precejšnjo razširitev za FDDI zanimivih okolij.

Tudi cene bodo z naraščajočim številom proizvajalcev FDDI-opreme padle. FDDI bo postal zanimiv najprej za večje ustanove ali skupine ustanov, ki bi rade svoje kapacitete povezale v zanesljivo in hitro mrežo. Veliko je namreč aparatur, ki so predrage, da bi si jih lahko privoščil vsakdo. Lahko pa jih učinkovito uporablja vsakdo, ki ima zagotovljen hiter dostop do njih. Tega pa si brez hitre in zanesljive mreže, kakršna je FDDI, ni možno predstavljati.

Literatura:

[1] Computer Networks, str. 166-168, Prentice Hall, 1988
[2] FDDI:Chapter Two, Data Communications - Lan Strategies, september 1991
[3] Smart Hub Vendors Move In on FDDI, Data Communications, oktober 1991
[4] MFS Makes a MAN Out of FDDI, Data Communications, oktober 1991
[5] Connectivity, The Sum of Its Parts, Byte, November 1991
[6] FDDI:Technology Brief, Network General, 1991
[7] Introduction to FDDI, Memo Instructional, Cabletron Systems, 1991

ZASNOVA INTEGRIRANEGA INFORMACIJSKEGA SISTEMA ZA PREPREČEVANJE ONESNAŽENJA

INFORMATICA 1/92

Keywords: information system model, factual database, reference database, relational database, prediction model, computer simulation, expert system, pollution prevention, pollution determination

S.A. Glažar, A. Kornhauser, R. Olbina, M. Vrtačnik
v sodelovanju z:
M. Ahčan, A. Cizerle-Belčič, D. Dolničar
Univerza v Ljubljani,
Fakulteta za naravoslovje in tehnologijo,
Kemijsko izobraževanje in informatika

Izvleček

V prispevku je predstavljen model integriranega informacijskega sistema za preprečevanje onesnaževanja. Sistem je sestavljen iz petih podsistemov, ki vključujejo vzroke in posledice onesnaževanja okolja. V prvem podsistemu so zajeti vzroki onesnaževanja (kvantitativni podatki o odpadkih in odpadnih vodah). Drugi podsistem omogoča napovedovanje količin odpadkov in odpadnih vod s simulacijo industrijske proizvodnje (predikcijski model). Tretji podsistem vključuje podatke o možnih načinih preprečevanja onesnaženja okolja z odpadki in odpadnimi vodami (referenčna baza podatkov o ravnanju z odpadki in odpadnimi vodami). Četrti podsistem podpira ugotavljanje učinkov onesnaževalcev v rečnih vodah, peti podsistem pa predstavlja ekspertni sistem za ugotavljanje in preprečevanje onesnaževanja rečnih vod.

Integrirani informacijski sistem naj bi vladnim ustanovam omogočal uveljavljanje učinkovitih strategij preprečevanja onesnaženja in ravnanja z odpadki.

Ključne besede: model informacijskega sistema, faktografska baza podatkov, referenčna baza podatkov, relacijska baza podatkov, predikcijski model, računalniška simulacija, ekspertni sistem, preprečevanje onesnaževanja, kontrola onesnaževanja

A MODEL OF AN INTEGRATED INFORMATION SYSTEM FOR POLLUTION PREVENTION

Abstract

A model of an integrated information system for pollution prevention is presented. It consists of five subsystems dealing with the causes and consequences of environmental pollution. The first subsystem presents pollution causes (quantitative data on waste and wastewaters generation). The second subsystem presents quantitative determination of waste and wastewater generation by simulation of industrial production and a waste generation model (prediction model of waste generation). The third subsystem includes possible ways of elimination of pollution caused by waste and wastewater generation (reference database on waste and wastewater management). The fourth subsystem is dedicated to determination of the consequences of pollution impacts on river waters, while the fifth subsystem presents the implementation of an expert system for river water pollution determination and prevention.

Such an integrated information system could provide a governmental institution with a powerful tool for implementation of pollution prevention and efficient waste management/ reduction strategy.

Uvod

V Sloveniji se kljub dokaj ostrim predpisom odpadki, ki vsebujejo nevarne snovi, še vedno kopičijo, ali pa neustrezno ter večkrat celo povsem neodgovorno odlagajajo. Posledice takih razmer se prav tako kopičijo, ne le v nepreglednih zalogah sodov z odpadki na tovarniških dvoriščih ter neustreznih deponijah, temveč tudi v izrednem slabšanju kvalitete zraka, vode in zemlje. To ima nujno škodljive posledice za ljudi, živali in rastline. Pri tem so občasne ekološke afere le vrh ledene gore, glavnina pa nosi dolgoročne posledice.

Bistveni pogoj za minimizacijo odpadkov ter njihovo pravilno predelavo in varno odlaganje je "rojstni list" vsakega odpadka, ki ga mora spremljati na celoviti poti. Ob upoštevanju stotin različnih odpadkov in desetin podatkov za vsakega ter nuji po sprotnem zasledovanju vseh sprememb je jasno, da sistem ravnanja z odpadki že v osnovi potrebuje integriran računalniško vodeni informacijsko - opozorilni sistem. Učinkovitost tega sistema se mora odražati v kompletnosti, preciznosti in ažurnosti ter v direktnem usmerjanju v akcije za preprečevanje nastajanja odpadkov in predelavo ter varno odlaganje.

V naporih za uvajanje celovitejših pristopov k reševanju problemov posebnih odpadkov in odpadnih vod ter njihovega vpliva na okolje je bil zasnovan integrirani informacijski sistem, ki vključuje pet ključnih podsistemskih enot:

I. Informacijsko nadzorni sistem za ravnanje s posebnimi odpadki,

II. Predikcijski model za napovedovanje količin odpadkov po vrstah industrijskih dejavnosti,

III. Referenčne baze podatkov o ravnanju s posebnimi odpadki in odpadnimi vodami,

IV. Relacijska baza podatkov o onesnaževanju rek,

V. Ekspertni sistem za nadzor in preprečevanje onesnaževanja rek.

Na shemi je podan model integriranega informacijskega sistema za preprečevanje onesnaževanja okolja z okvirnimi strukturami posameznih podsistemov ter njihovimi povezavami.

I. Struktura informacijsko - nadzornega sistema za ravnanje s posebnimi odpadki

Informacijsko-nadzorni sistem je bil zasnovan leta 1988 na Fakulteti za naravoslovje in tehnologijo (Kemijsko izobraževanje in informatika) na osnovi podatkov iz popisnih listov, ki so bili poslani vsem potencialnim proizvajalcem posebnih odpadkov v Sloveniji in obiskov pri imetnikih odpadkov. Obiske in zbiranje popisnih listov je izvedel Kemijski inštitut Boris Kidrič v letih 1988-89 v 12 slovenskih regijah: pomurski, mariborski, koroški, celjski, zasavsko - revirski, posavski, dolenjski, ljubljanski, kraško - notranjski, gorenjski, severno-primorski in obalno - kraški regiji. To omogoča spremljanje časovnih trendov ter prostorskih razporeditev posameznih vrst odpadkov po količini in načinu vračanja v okolje v posameznih regijah in znotraj njih v posameznih občinah. Sistem stalno dopolnjujemo s podatki za tekoče leto, ki so zbrani s pomočjo Republiškega sanitarnega inšpektorata na osnovi najave odpadkov po Pravilniku za ravnanje z odpadki, ki vsebujejo nevarne snovi.

Informacijsko-nadzorni sistem je zgrajen s programskim paketom dBase III+ in programskim jezikom Clipper na IBM kompatibilnem mikroračunalniku kot relacijski sistem petih med seboj povezanih baz. Prehodi med bazami so možni prek matičnih številk DO ter prek ključnih številk odpadkov.

Prvi modul vsebuje statistične podatke o regijah, občinah, proizvajalcih odpadkov in njihovih dejavnostih. Vsak povzročitelj posebnih odpadkov je identificiran s statistično matično številko DO. Šifra dejavnosti DO omogoča obdelavo podatkov glede na nastajanje oz. ravnanje z odpadki v posameznih dejavnostih.

Prostorsko je lokacija DO opredeljena z naslovom in geografskimi koordinatami po Gauss-Kruegerju, kar omogoča grafične prikaze lociranosti oz. dispergiranosti posameznih vrst odpadkov, izračun njihovih težišč nastajanja, povezave s prostorskimi informacijskimi sistemi, izdelavo kart onesnaženja itd. Cona varstvenega pasu vodnih virov omogoča oceno ogroženosti posameznih vodonosnih področij.

Drugi modul je šifrant posebnih odpadkov in vključuje naslednje karakteristike o odpadku iz Pravilnika o ravnanju s posebnimi odpadki, ki vsebujejo nevarne snovi:

(1) ključna številka odpadka po katalogu, (2) ime in vrsta odpadka, (3) vrsta nevarnosti, (4) priporočena obdelava oz. odlaganje v okolje.

Tretji modul vsebuje podatke iz popisa odpadkov. Razdeljen je na segment s kvalitativnimi in segment s kvantitativnimi podatki o odpadku. V segmentu s kvalitativnimi podatki so: (1) ključna številka odpadka po katalogu Pravilnika, (2) okvirna sestava odpadka, (3) konsistenca odpadka, (4) delež vode, (5) pH, barva, vonj. Segment s kvantitativnimi podatki pa vsebuje podatke o količini odpadka v tekočem letu ter zalogi odpadka na dan popisa. Merske enote za podajanje količine odpadkov so lahko tone in/ali m^3.
Četrti modul podaja ravnanje z odpadkom od nastanka do ponovnega vračanja v okolje. V tem sklopu zajete podatke lahko razdelimo v štiri skupine: (1) podatki, ki se nanašajo na mesto nastanka: ravnanje z odpadkom na mestu nastanka in način začasnega shranjevanja; (2) podatki o zbiranju in transportu odpadkov: način in izvajalec odvoza, frekvenca odvoza; (3) podatki o procesiranju odpadkov: vrsta procesiranja, izvajalec procesiranja; (4) podatki o končnem odlaganju v okolje za odpadke in produkte pri njihovem procesiranju: način odlaganja odpadka, izvajalec odlaganja.

Peti modul podpira kontrolo podatkov o prijavljenih odpadkih in je hkrati osnova za gradnjo modela za predikcijo nastajanja odpadkov. Baza zajema: (1) kode dejavnosti (ISIC- International Standard Industry Code) z značilnimi odpadki ter (2) kvalitativne podatke o porabljenih surovinah in nastalih produktih za posamezne industrijske veje znotraj kode dejavnosti.

II. Referenčna baza podatkov o ravnanju s posebnimi odpadki in odpadnimi vodami

Podpora za reševanje problemov odpadkov, zajetih v informacijsko-nadzornem sistemu je referenčna baza, ki vključuje: (1) specializirano bibliografsko bazo "Waste and Wastewater Processing Technology", ki je od leta 1989 dosegljiva tudi online prek Računalniškega centra Univerze v Mariboru in vključuje blizu štiritisoč dokumentov (maj 1991), (2) referalno bazo "Waste Management Experts" in (3) referalno bazo "Waste Management Equipment Producers".

III. Predikcijski model

Struktura informacijsko - nadzornega sistema za ravnanje z odpadki tvori osnovo za opredeljevanje ključnih parametrov industrijske proizvodnje, pri kateri nastajajo posebni odpadki. Podatki petega dela informacijsko-nadzornega sistema so bili uporabljeni za razvoj matematičnega modela, ki omogoča kvantitativno napoved nastajanja odpadkov. Matematični model je v fazi dograjevanja in testiranja za štiri postopke v kemijski industriji.

Predikcijski model je hkrati tudi osnova za optimizacijo postopkov v industrijski proizvodnji, ki vključuje tudi "cost-benefit" analize. Model podpira tudi kvantifikacijo tistih proizvodnih stroškov, ki so posledica emisij onesnaževalcev v okolje.

Matematični model je preizkušen na dveh računalniških paketih, STELLA Dynamic Simulation Software in EXCEL. Oba paketa omogočata simulacijo sistema z enostavnim spreminjanjem vrednosti parametrov sistema. Na ta način podpirata predikcijo nastajanja odpadkov in/ali optimizacijo industrijske proizvodnje.

Osnovni cilj simulacije je uvajanje takšnih industrijskih pogojev, pri katerih nastaja

najmanjša količina odpadkov. Preprečevanje onesnaženja v industrijski proizvodnji je visoka prioriteta na področju ravnanja z odpadki v industrijsko najbolj razvitih deželah Evrope in ZDA. "Bolje preprečevati kakor zdraviti" velja tudi na področju odpadkov.

IV. Relacijska baza podatkov o onesnaževanju rek

Relacijska baza ima dva ključna segmenta:

- **splošni segment:** "Prednostni vodni onesnaževalci",
- **specifični segment:** "Onesnaženje slovenskih vodotokov ter potencialni izvori onesnaženja"

Splošni segment: "**Prednostni vodni onesnaževalci**"

Kot izhodišče za izbor vodnih onesnaževalcev je služil Zakon o nadzoru nad onesnaževanjem vode ZDA (U.S. Water Pollution Act), ki v 311. členu navaja 299 spojin-potencialnih vodnih onesnaževalcev. Onesnaževalci so opredeljeni na osnovi dveh ključnih kriterijev: toksičnosti in stabilnosti v vodi.

Za zasnovo splošnega segmenta relacijske baze so bile iz seznama 299 spojin izbrane najprej tiste, ki so opredeljene kot **glavni (priority pollutants)** vodni onesnaževalci. Seznam vključuje 129 spojin.

Splošni segment baze je zgrajen iz **sedmih enot**, ki so povezane z imenom onesnaževalca:

(1) klasifikacija in osnovne fizikalno-kemijske lastnosti onesnaževalcev,

(2) sinonimi in standardne kode onesnaževalcev,

(3) standardi in faktorji onesnaževanja,

(4) učinki onesnaževalcev na vodne organizme,

(5) možne interakcije onesnaževalcev z vodo,

(6) industrije - potencialni izvori onesnaževalcev.Ta enota vključuje numerične podatke o maksimalnih koncentracijah onesnaževalcev v industrijskih odpadnih vodah za 24 osnovnih industrijskih panog. (Podatki so bili prenešeni iz ameriške baze RREL, ki jo gradi Risk Reduction Engineering Laboratory, U.S. EPA, Cincinnati, Ohio, ZDA),

(7) kemijske analizne metode za določanje različnih koncentracij onesnaževalcev v vodi.

Relacijski model je zasnovan tako, da podpira odkrivanje parcialnih odnosov med podatki znotraj vsake posamezne enote baze, ter celovitejših odnosov z zasnovo poljubnih povezav med podatki iz različnih enot baze. Ta je osnova za prepoznavanje vzorcev kompleksnih vplivov onesnaževalcev na okolje.

Specifični segment: "**Onesnaženje slovenskih vodotokov ter potencialni izvori onesnaženja**"

Specifični segment baze vključuje tri sklope podatkov:

(1) **register slovenskih rek** in merilnih mest,

(2) **register industrijskih obratov**, ki so locirani ob rekah, s kodami njihove dejavnosti,

(3) **rezultate rednih meritev** osnovnih fizikalno-kemijskih in bioloških pokazateljev onesnaženja slovenskih rek.

Register slovenskih rek in merilnih mest je prek kode merilnega mesta povezan s serijo meritev pokazateljev stopnje onesnaženja za izbrano merilno mesto. Koda reke pa povezuje register rek in merilnih mest z registrom industrijskih obratov. Zadnji bo prek kode industrijske dejavnosti povezan z maksimalnimi koncentracijami in vrstami onesnaževalcev v nepredelani industrijski odpadni vodi za izbrano industrijsko panogo. (Ta del baze je zasnovan le modelno, ker FNT-KII trenutno ne razpolaga z rezultati analiz odpadnih vod slovenske industrije.)

V. Zasnova modela baze znanja za preprečevanje vzrokov onesnaženja slovenskih vodotokov

Podatki v obeh relacijskih bazah "Prednostni vodni onesnaževalci" in "Onesnaženje slovenskih vodotokov ter potencialni izvori onesnaženja" predstavljajo na nivoju ekspertnega sistema deklarativno znanje. Relacije, ki jih modeli podpirajo, pa omogočajo prepoznavanje elementov proce-duralnega znanja, ki je bistveni sestavni del baze znanja ekspertnega sistema. Ker je izbrano področje izrazito interdisciplinarno in multiparametersko, ni mogoče vseh relacij med parametri razložiti z globokim znanjem področja, lahko pa na osnovi primerov v deklarativnem delu baze izpeljemo izkustvena pravila. Stopnja možne posplošitve pravila zavisi od števila primerov, ki pravilo ponazarjajo.

Uporabniški vmesnik baze znanja je razvit tako, da omogoča preverjanje dveh ključnih hipotez o možnih vzrokih onesnaženja slovenskih vodotokov:

- **hipoteza 1:**
  vzroki onesnaženja so industrijski odpadki ali/in nečiščene industrijske odpadne vode, ki jih spuščajo v bližnjo reko;

- **hipoteza 2:**
  vzrok onesnaženja je neustrezno ravnanje s surovinami, ki so bile slučajno ali namensko odvržene v bližnjo reko.

Logične povezave, ki omogočajo preverjanje prve hipoteze, so že razvite in podprte z bazami podatkov ter implementirane na lupini ekspertnega sistema KnowledgePro.

Integrirani informacijski sistem za preprečevanje onesnaževanja okolja je poskus zasnove celovitejšega pristopa pri reševanju problemov odpadkov in odpadnih vod v Sloveniji. Sistem nudi podporo za:

(1) kvalitativno, kvantitativno in geografsko opredeljevanje nastajanja odpadkov in odpadnih vod,

(2) zasledovanje trendov na področju novih tehnologij ravnanja z odpadki in odpadnimi vodami ter študij možnosti njihovega uvajanja,

(3 predikcijo količin odpadkov za posamezne industrijske veje,

(4) optimizacijo industrijske proizvodnje z vidika nastajanja odpadkov,

(5) zasledovanje stopnje onesnaženja slovenskih rek,

(6) identifikacijo najbolj verjetnih izvorov onesnaženja rek,

(7) zmanjševanje onesnaženja voda z uvajanje ustreznih tehnik čiščenja odpadnih vod in s tem zmanjševanje vplivov onesnaženja na vodne ekološke sisteme,

(8) identifikacijo področij rečnih izlivov, kjer bi bilo potrebno uvesti ukrepe za zmanjšanje onesnaženja celovitega rečnega toka,

(9) ukrepanje v primerih onesnaženja rek za optimalno zaščito potencialnih izvorov pitne vode.

Bibliografija

Glažar, S.A., Grilc, V., Kornhauser, A.: Odpadne snovi v okolju in ukrepi za zmanjšanje njihovih številnih škodljivih učinkov. Raziskovalno-razvojna naloga, Raziskovalna skupnost Slovenije, PORS 21-2813, 1988.

Glažar, S.A., Grilc, V., Husić, M.: Problem posebnih odpadkov v Sloveniji, UJMA, 4, str. 120-124, 1990

Glažar, S.A., Kornhauser, A., Olbina, R.: Waste Management Information System: Its Role in Pollution Prevention and Introducing Cleaner Technologies and Products, (Vabljena sekcijska predstavitev na "International Conference on Pollution Prevention: Clean Technologies and Clean

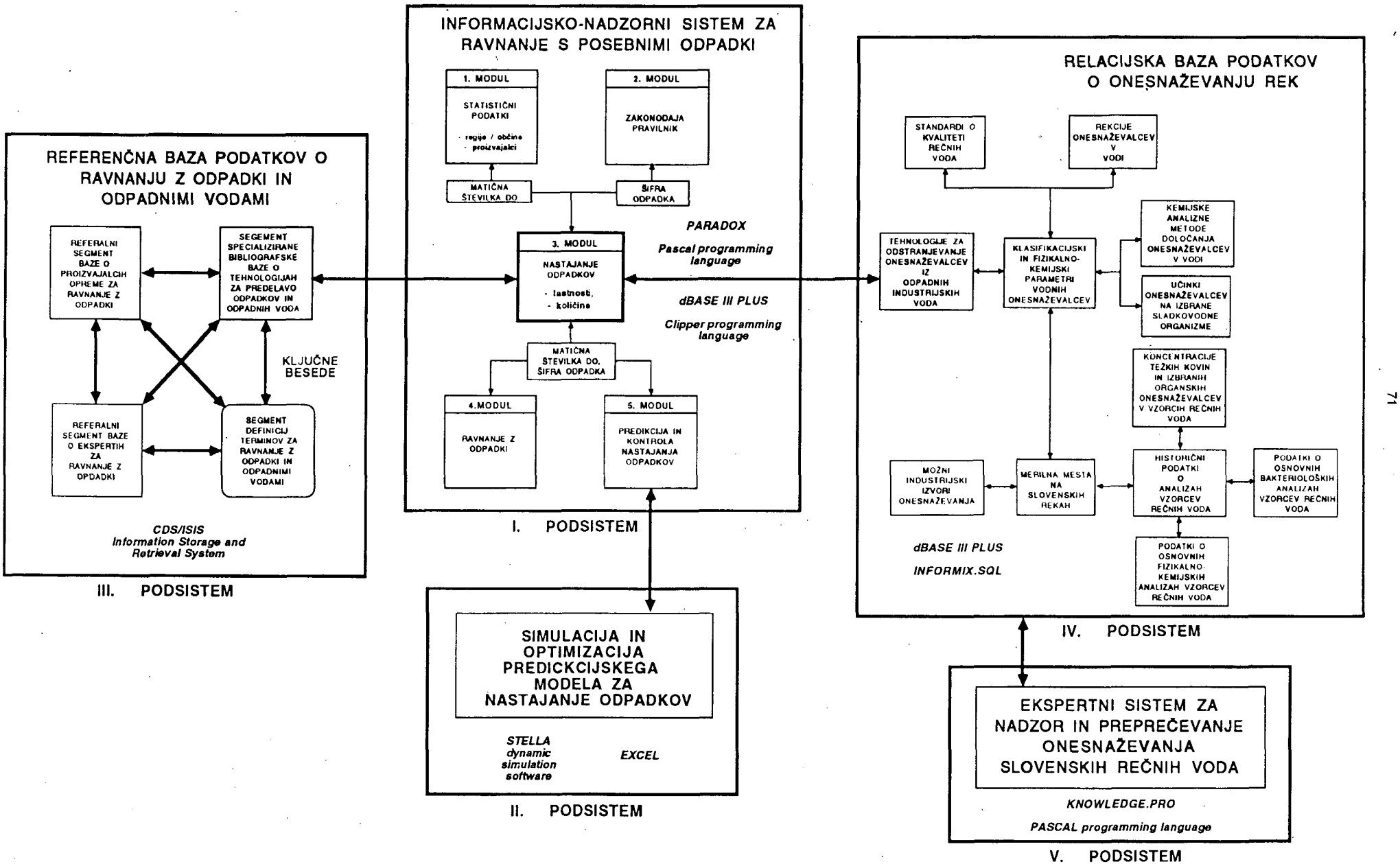
Products", Washington D.C., June 10 - 13, 1990)

Olbina, R.: Računalniško podprto modeliranje ravnanja s nevarnimi odpadki, Disertacija, 271 str., 1991

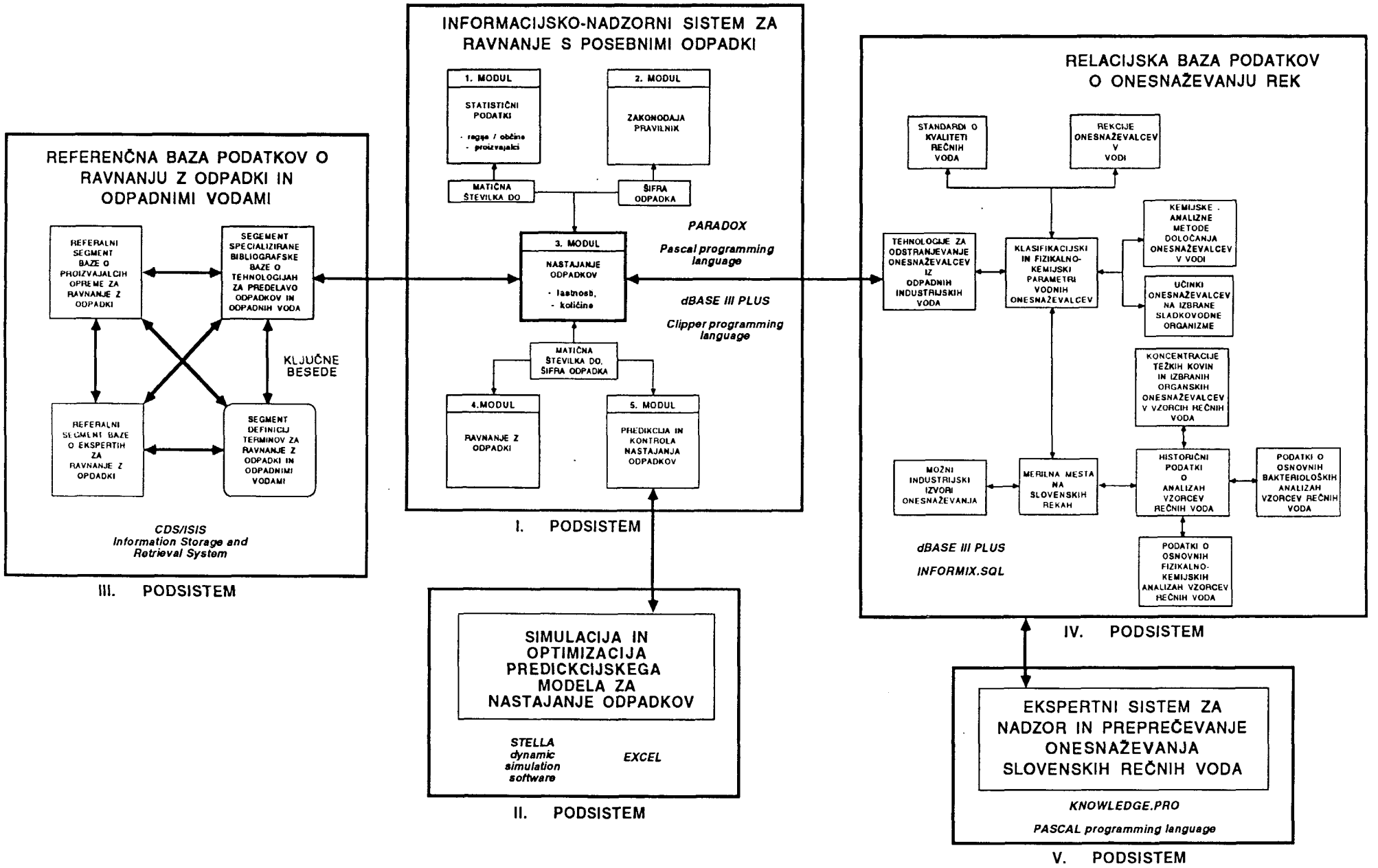
Vrtačnik, M., Dolničar, D. Čok, P.: Development of Computerized System for Automatic Assignment of River Water Pollution Levels, (Vabljeno sekcijsko predavanje na 4th Asian Chemical Congress, Regional Seminar on Chemical Information Network, Beijing, China, August 26 - 30, 1991).

Vrtačnik, M., Dolničar, D., Cizerle, A., Čok, P., Glažar, S.A., Olbina, R.: Design of an Expert System for Water Pollution Determination / Prevention, 15 str., (Članek sprejet v objavo v reviji Expert Systems with Applications), 1991

SHEMA MODELA
INTEGRIRANEGA INFORMACIJSKEGA SISTEMA ZA PREPREČEVANJE ONESNAŽEVANJA OKOLJA



SHEMA MODELA
INTEGRIRANEGA INFORMACIJSKEGA SISTEMA ZA PREPREČEVANJE ONESNAŽEVANJA OKOLJA



UPORABA YOURDONOVE STRUKTURNE METODE V IDEJNEMU PROJEKTU PROCESNEGA VODENJA PRI ANALIZI OBSTOJEČEGA STANJA

INFORMATICA 1/92

Keywords: CASE, feasibility study, system analysis, process control

Marjan Rihar
Inštitut Jožef Stefan, Ljubljana

POVZETEK

V uvodnem delu članka so na kratko nakazane možnosti uporabe različnih računalniških orodij v idejnih projektih. Nadalje so podani mesto, vsebina in cilji študije uresničljivosti ter njena primerjava z idejnim projektom. Sledi predstavitev sistemskega pristopa k izdelavi modela obstoječega stanja industrijskega sistema, kot sestavnega dela idejnega projekta. Pristop temelji na Yourdon-ovi strukturni metodi za sistemsko analizo. Model obstoječega stanja je izdelan s podporo računalniškega orodja Analyst/Designer Toolkit. Celoten model sestoji iz informacijskega in materialno-energijskega modela. Informacijski model je obdelan s tehniko dekompozicije industrijskega sistema na organizacijske enote in dalje na tehnološke podprocese, kjer je poudarek na modeliranju enostavnega vodenja. Materialno-energijski model prikazuje organizacijsko enoto proizvodnjo kot kontekstni proces. Ta je nadalje razčlenjen in prikazan na sistemskem diagramu materialnih in energijskih tokov med tehnološkimi podprocesi in njihovo okolico. Oba modela vsebujeta še slovar podatkov, procesne specifikacije pa so skupne. Celovit pregled nad obstoječim sistemom zagotavlja konsistentna povezava med modeloma. Članek je osredotočen predvsem na uporabo sistemskega pristopa v idejnih projektih, ki predvidevajo posodobitev obstoječega stanja z uvajanjem računalniškega vodenja.

THE USAGE OF YOURDON STRUCTURED METHOD IN THE INITIAL PROJECT STUDY OF PROCESS CONTROL FOR CURRENT STATE ANALYSIS. In the introductory part of this article, the possibilities of usage of the various computer tools for creating initial project study are showed briefly. Further, the motive, contents and goal of feasibility study and its comparison with initial project study are given. Then, the presentation of system approach to building the industrial system current state model as part of initial project study follows. The approach is based on Yourdon structured method for system analysis. The current state model is built up with support of Analyst/Designer Toolkit computer tool. The entire model consists of informational and material-energy model. The former one is done by decomposition of whole industrial system into industrial establishment organization units and further into technological subprocesses. Here, the main importance is given on modelling of basic process control. The latter model shows the organization unit manufacturing as context process. It is onward partitioned and depicted on system diagram of material and energy flows among the technological subprocesses and their environment. The both model still contain the private data dictionary, but the process specifications are common. The complex view over existing industrial system is achieved by the consistent coupling of models. The paper is mainly concentrated on application of system approach in the initial project studies concerned on enhancing the system current state by means of computer control.

1. UVOD

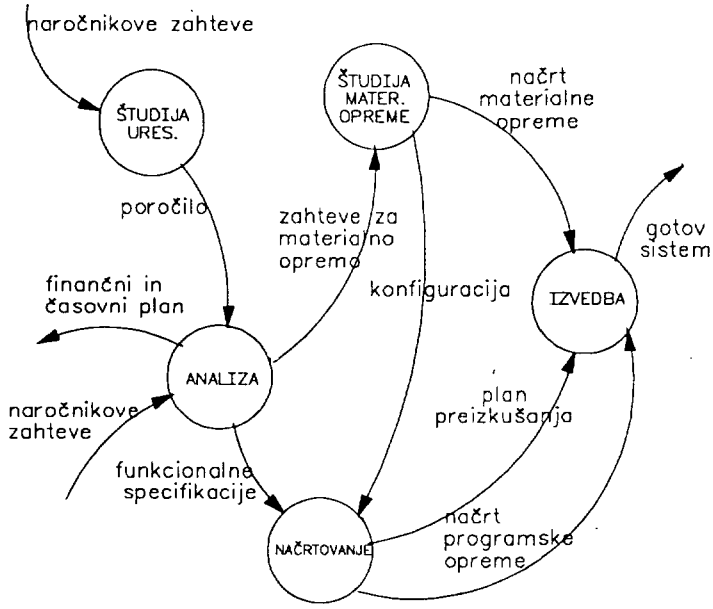
Ideje za zgraditev vsakega novega ali razširitev obstoječega nenaravnega sistema se morajo vklapljati v okolje sistema oziroma izhajati iz obstoječega stanja sistema. Formalno so tako dobljena spoznanja in nove deje predstavljene z *idejnim projektom*. V našem prostoru se je uveljavilo načelo, da je idejni projekt prvi dokument v okviru projektne naloge (Šube91). V idejnem projektu so predstavljene vsebinske ideje projektne naloge, cilji projekta, organizacijska in izvedbena priporočila investitorju za izvajanje projekta, utemeljitev upravičenosti projekta ter merila za zagotavljanje kvalitete vsebine in kvalitete vodenja projekta. Raznovrstnost področij narekuje številne različne pristope in tehnike izdelave idejnih projektov, ki so večinoma plod večletnih izkušenj.

V dobi, ko je postal računalnik splošno priznano in uporabno orodje, je nujno ali pa vsaj racionalno, da ga v največji meri izrabimo tudi na tem področju. Na razpolago imamo kar nekaj za te namene primernih računalniško podprtih orodij. To so npr. programski paketi za risanje (ACAD, ...), urejevalniki besedil (WordPerfect, WordStar, ...), programski paketi za obdelavo podatkov (LOTUS, DBASE, ...), paketi za pomoč pri planiranju (SuperProject, ...), itd. V zadnjem desetletju so se v svetu in tudi pri nas pojavila orodja CASE. Ker sta sestavna dela idejnega projekta tudi *analiza obstoječega stanja* in preliminarni načrt funkcij bodočega sistema, kjer pride do izraza uporaba tehnik funkcionalne dekompozicije, se je izkazalo, da ta orodja lahko v znatni meri pripomorejo k hitrejši, enostavnejši in doslednejši izvedbi le teh. Za uporabo v *idejnih projektih vodenja industrijskih procesov* (nadalje *procesnega vodenja*) so primerna vsa tista "front-end" orodja CASE, ki pokrivajo vsaj fazo sistemske analize in podpirajo "real-time" razširitve konvencionalnih, na pretok podatkov orientiranih metod. Med precej razširjena in relativno cenena tovrstna orodja spadajo npr. *Software Egineerig Workbench* proizvajalca *Yourdon Inc.*, *Select* proizvajalca *ISS*, *System Architect* proizvajalca *Popkins*. Dražja orodja imajo pogosto že vgrajene mehanizme za upravljanje s projekti (npr. *Excelerator* proizvajalca *Index Technology Inc.*), vendar zaradi visoke cene v našem prostoru širše niso posebno zanimiva. Vsa našteta orodja omogočajo izvedbo faze sistemske analize s pomočjo uveljavljenih in preiskušenih metod programskega oziroma sistemskega inženirstva. Z izbiro primernih (kar zavisi od področja in vsebine idejnega projekta) splošnonamenskih računalniških orodij in orodij CASE, lahko torej ustvarimo relativno ceneno in konsistentno okolje za izdelavo idejnih projektov.

V nadaljevanju članka bomo prikazali uporabo Yourdon-ove strukturne metode, podprte z orodjem Analyst/Designer Toolkit (Analyst/Designer Toolkit je del paketa Software Engineering Workbench), v zelo ozkem segmentu idejnega projekta, to je v analizi obstoječega stanja. Pri tem je potrebno poudariti, da smemo za analizo obstoječega stanja potrošiti le toliko resursov (časa, denarja, moštva, ...), *kolikor je nujno za osnovno razumevanje obstoječega sistema*, to je toliko, da lahko na osnovi te analize zgradimo model novega sistema.

2. IDEJNI PROJEKT IN "FEASABILITY STUDY"

Ena izmed zelo razširjenih in splošno uporabnih metod za strukturno analizo je Yourdon-ova strukturna metoda (Your89). Za projekte procesnega vodenja je še posebej primerna, ker omogoča modeliranje funkcionalnih zahtev sistemov z diagrami tokov podatkov (Data Flow Diagrams - DFD), odnosov med podatki z diagrami razmerij med entitetami (Entity Relationship Diagrams - ERD) in časovno obnašanje sistemov z diagrami prehajanja stanj (State Transition Diagrams - STD). Diagrami tokov podatkov so načeloma podobni blok shemam v avtomatiki, diagrami prehajanja stanj pa so dejansko diagrami prehajanja stanj, ki jih poznamo s teorije avtomatov. Yourdon-ovo strukturno metodo podpira večina orodij CASE.



Slika 1 Strukturni življenski krog sistemov

Yordon-ova strukturna metoda obravnava faze življenja sistemov v okviru takoimenovanega *strukturnega življenjskega kroga.* Te faze so:

- študija uresničljivosti (feasibility study),
- strukturna analiza,
- strukturno načrtovanje,
- izvedba in
- vzdrževanje.

Strukturni življenjski krog smatra posamezne faze kot paralelne (sočasne) procese (Slika 1). Projekt zgraditve celotnega sistema je sestavljen iz vrste povezanih podprojektov, od katerih vsak obravnava določeno fazo. Rezultati obravnave so zbrani v odgovarjajoči projektni dokumentaciji - izhodnem dokumentu.

V kontekstu tega članka nas predvsem zanima faza *študija uresničljivosti.* Poglavitna vprašanja, na katera naj bi študija dala odgovora, sta:

- ZAKAJ bo namenjen sistem, katerega osnovne poteze bomo začrtali s to študijo in
- ALI je sistem sploh smiselno zgraditi (kar je posebno pomembno pri zelo obsežnih sistemih).

Yourdon-ova metoda skuša poiskati odgovor na vprašanji z izgradnjo in analizo naslednjih modelov, ki naj bi bili sestavni deli študije uresničljivosti:

- izvedbeni model obstoječega stanja,
- osnovni model obstoječega stanja,
- preliminarni vsebinski model - idejna zasnova - bodočega sistema (popolnoma novega ali dopolnjenega starega) in
- preliminarni cenovni model.

S pomočjo naštetih modelov so uresničeni cilji študije, ki so:

- ugotovitev problemov naročnika (bodočega uporabnika) ter možnosti reševanja le teh,
- ugotovitev začetnih (preliminarnih) zahtev naročnika,
- ugotovitev in ovrednotenje preliminarnih možnih rešitev,
- odločitev za izbiro konkretne rešitve problemov,
- izdelava poročila za tehnično in poslovno osebje naročnika,
- predlaganje izvajalca naslednjih faz projekta in
- predlaganje virov in načinov financiranja projekta.

Cilji študije so podani v *poročilu* (feasibility report). Vsebina poročila naj bi obsegala: problematiko dosedanjega stanja sistema, opredelitev ciljev novega sistema, idejne rešitve, ekonomsko analizo, povzetek, priporočila in dodatke. Če primerjamo namen, način izdelave in vsebino tega *poročila* z namenom, načinom izdelave in vsebino *idejnega projekta* (Čern91), vidimo, da med njima ni bistvene vsebinske razlike. Zaključimo lahko, da ima idejni projekt v domačem okolju zelo podoben namen in pomen, kot ga ima v svetu študija uresničljivosti. V idejnem projektu lahko torej uporabimo enake metode in na enak način kot v študiji uresničljivosti. To prinaša vrsto prednosti, ki se kažejo v medsebojni primerljivosti različnih idejnih projektov, v možni podpori računalniških orodij in v postavljeni vsebinski in metodološki osnovi za izvajanje nadaljnjih faz projekta.

3. SISTEMSKI PRISTOP

Yourdon-ova *metodologija*, ki obsega skladno povezavo strukturnih *metod* v zgodnjih življenskih fazah sistema, je najbolj uporabna na področju programskega inženirstva. Problemi, ki se pojavljajo pri projektih procesnega vodenja, pa so širši in jih je potrebno reševati v okviru sistemskega inženirstva. Izkazalo se je, da je Yourdon-ova metodologija primerna tudi za uporabo na tem področju (Čern90, Rih91b).

K vsakemu projektu procesnega vodenja moramo pristopiti sistemsko, to pa pomeni, da moramo upoštevati značilnosti sistemov znanih iz sistemske teorije. Ena izmed značilnosti sistemov je, da v vsakem sistemu obstoji množica relacij med *energijo*, *snovjo* in *informacijami* (Bowl81). Če hočemo sistem spoznati, in prav to hočemo v *začetni fazi* idejnega projekta, moramo poiskati navedene relacije. Če hočemo sistem dopolniti ali/in spremeniti, moramo dopolniti ali/in spremeniti tudi relacije, kar je naloga *naslednje faze* idejnega projekta (idejne zasnove novega sistema). Glede na to, na katere vrste relacij želimo vplivati, govorimo o:

- avtomatizaciji in racionalizaciji, kjer želimo vplivati na izvajanje funkcij tehnoloških podprocesov, s tem pa tudi na racionalno izrabo energije in surovin in
- informatizaciji, kjer želimo vplivati na kvaliteto in razširjenost podatkovnih povezav.

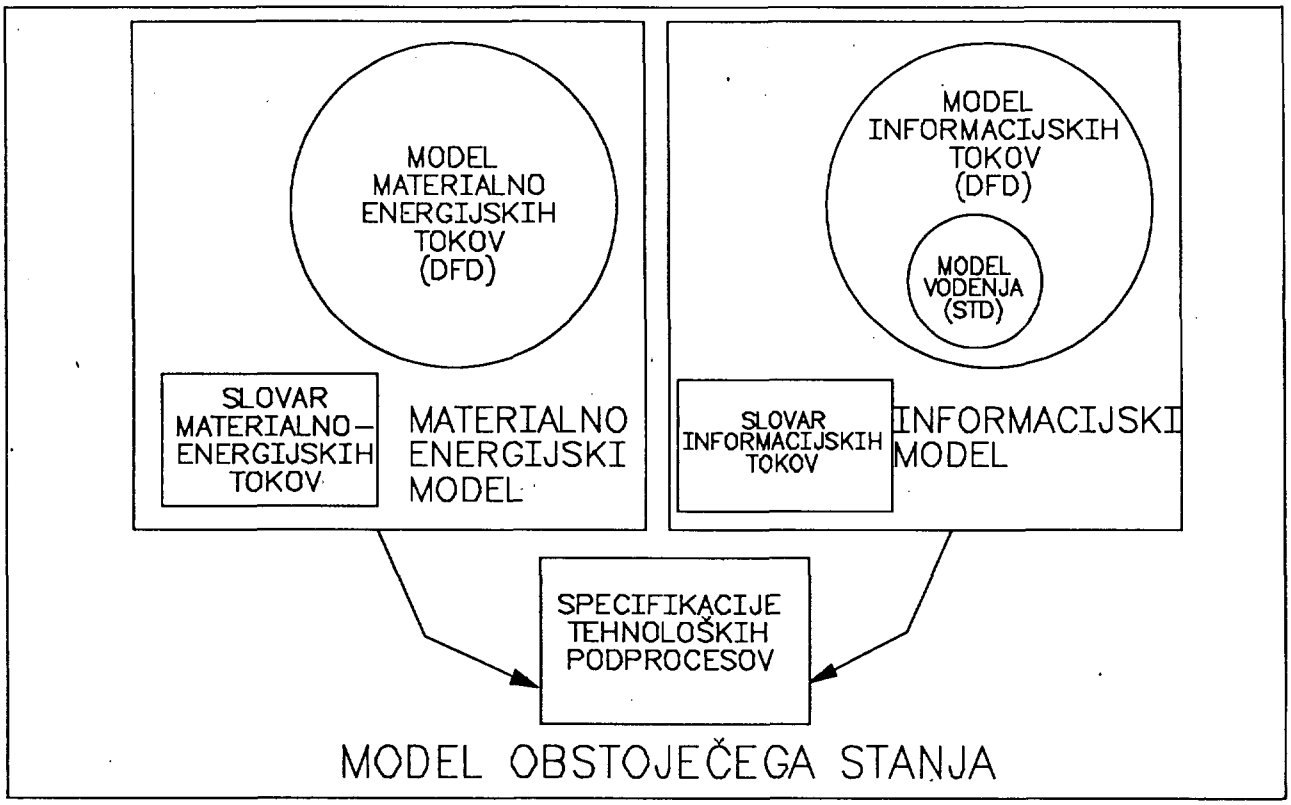
Na sistem vplivamo z vodenjem energije, snovi in informacij, katerega idejno zasnovo nakažemo v idejnem projektu.

Yourdon-ova strukturna metoda za analizo z možnostjo "top-down" ali "bottom-up" pristopa je zelo primerna za zgraditev modelov vseh treh komponent sistemov v idejnem projektu. Posebno pomembno je dejstvo, da izdelavo teh modelov podpirajo računalniška (CASE) orodja. Tako lahko z uporabo teh orodij precej povečamo hitrost in kvaliteto izvedbe delov idejnega projekta.

4. ANALIZA OBSTOJEČEGA STANJA

V vsakem primeru, pa naj bo idejni projekt namenjen izboljšanju že obstoječega sistema ali graditvi popolnoma novega, mora *izvajalec idejnega projekta* (dalje izvajalec) dovolj spoznati sistem. Formalno gledano je izvajalec sistemski analitik, dejansko pa oseba, ki ima znanje s panoge, ki ji pripada sistem, metodološko znanje s področja zbiranja informacij, znanje avtomatike, računalništva in znanje iz vodenja projektov. Izvajalec s pomočjo različnih tehnik zbiranja informacij črpa od *predstavnika naročnika* (dalje naročnik) znanje o sistemu. Na podlagi informacij gradi model obstoječega stanja. Ker izvajalec in naročnik tesno sodelujeta, se tudi naročnik postopoma uvede v metodološko vodeni pristop tako, da čez nekaj časa lahko povsem samostojno sodeluje pri izdelavi modela.

V idejnem projektu razgradimo sistem le toliko, kolikor je nujno potrebno za razumevanje njegovih funkcij. Rezultat razgrajevanja (analize),



Slika 2 Model obstoječega stanja

ki je plod skupnega dela izvajalca in naročnika, je:

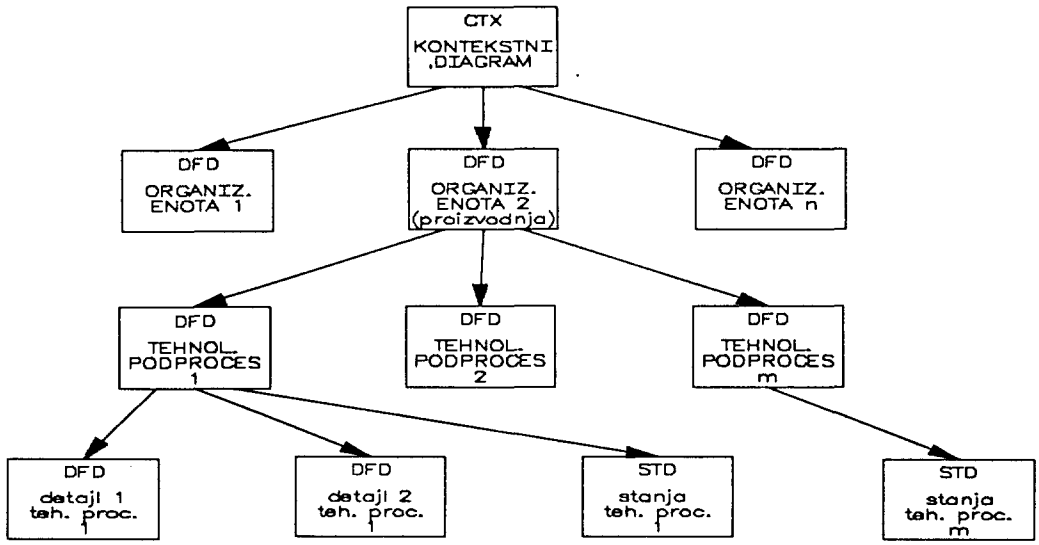
- informacijski model, ki vsebuje:
  - informacijski preliminarni osnovni model - v primeru graditve popolnoma novega sistema ali
  - informacijski izvedbeni model - v primeru izboljševanja obstoječega sistema in

- materialno-energijski model.

Informacijski izvedbeni model je potrebno pretvoriti v preliminarni osnovni model in sicer z odstranitvijo vseh izvedbenih podrobnosti. Preliminarni osnovni model je torej model funkcij in kot tak služi za dopolnjevanje z vsemi predvidenimi novimi funkcijami v nadaljevanju projekta.

Za celoten pregled nad sistemom je izrednega pomena povezanost in *skladnost* med informacijskim in materialno-energijskim modelom. Pragmatično gledano, povezanost in skladnost lahko dosežemo z enakim načinom graditve, z enako strukturo in z enako predstavitvijo obeh modelov. Formalno eksakten način določitve povezanosti in skladnosti med modeloma pa je zahtevna naloga in precej presega obseg tega članka.
Naštete poznane metodološke rešitve, razpolaganje z orodjem CASE in izkušnje, pridobljene pri izdelavi številnih idejnih projektov, so bile osnova za naš pristop k analizi obstoječega stanja. Pristop smo metodološko izpopolnjevali sprotno z delom na konkretnem projektu (Rih91b). V nadaljevanju članka objavljamo njegove osnovne značilnosti.

Informacijski in *materialno-energijski* model je grajen po principu "top-down" z uporabo Yourdon-ove metode, nadgrajeno z grafično sintakso (Rih91a) in ob pomoči orodja Yourdon Analyst/Designer Toolkit. Struktura obeh modelov je enaka: model tokov, izdelan z uporabo diagramskih tehnik in slovar podatkov. Opisi tehnoloških podprocesov (procesne



Slika 3 Struktura modela informacijskih tokov

specifikacije), ki nastopajo kot procesi (procesi, kot elementi diagrama toka podatkov) na obeh modelih, so skupni in obsegajo opis transformacij podatkov, surovin in energije (Slika 2). Opisani pristop je možno uporabiti pri analizi obstoječega stanja v vseh podobnih sistemih.

4.1 Informacijski model

Informacijskih model prikazuje tokove obstoječih *informacij* med posameznimi obstoječimi (računalniško nepodprtimi) tehnološkimi podprocesi ter procesi upravljanja (vodenja). S pojmom informacije so mišljeni vsi tehnološki podatki v tehnoloških podprocesih, ustna in pisna poročila, avtomatska signalizacija in merjene veličine, dobljene z merilno procesno opremo.

Model informacijskih tokov ima hierarhično strukturo (Slika 3).

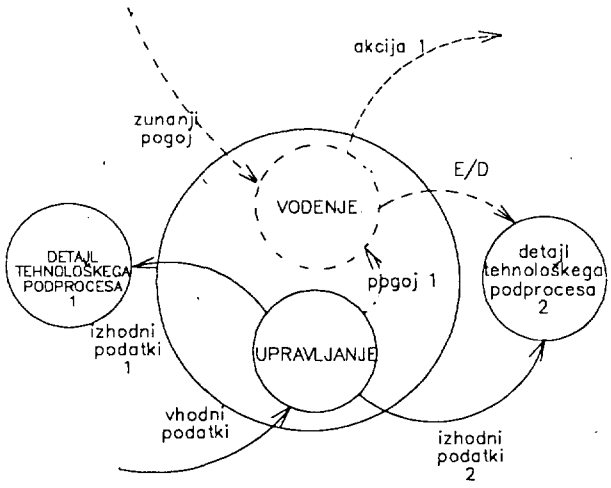
Na *kontekstnem nivoju* (CTX) je s kontekstnim diagramom tokov podatkov prikazan pretok globalnih informacijskih tokov na *nivoju tovarne*, med organizacijskimi enotami tovarne in okolico, ki jo smatramo kot danost in je nadalje ne analiziramo.

Na *naslednjem nivoju* so z diagrami tokov podatkov (DFD) prikazane dejavnosti *organizacijskih enot tovarne* (proizvodnja, merilnica, skladišče, ...). V odvisnosti od cilja in obsega idejnega projekta izberemo za nadaljnje razčlenjevanje ustrezno število organizacijskih enot. Za idejne projekte, ki vsebinsko pokrivajo *izboljšanje vodenja* tehnoloških podprocesov, je predvsem pomembna nadaljna razčlenitev organizacijske enote *proizvodnje*, kjer je potrebno ugotoviti tipične tehnološke podprocese.

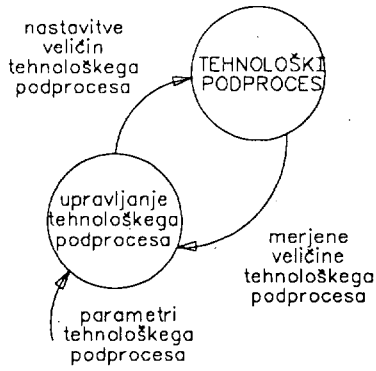
Na naslednjem *nižjem nivoju* so prikazani *tehnološki podprocesi*. Vsak tehnološki podproces je prikazan z diagramom tokov podatkov (DFD), ki pa je lahko še nadalje razčlenjen v detajle. Na vsakem diagramu najdemo poleg tipičnih funkcij tehnoloških podprocesov tudi procesa *vodenje* in *upravljanje* (Slika 4). *Dejansko je to en proces, katerega funkcija je transformacija podatkovnih in kontrolnih tokov*. Delitev na procesa vodenje in upravljanje je umetna (besedi sta sinonima) in smo jo uvedli le zaradi skladnosti s pravili razširjenih diagramov tokov podatkov za modeliranje zahtev realnega časa (Your89).

Procesa *vodenje* in *upravljanje* predstavljata:

- funkcije popolnega avtomatskega vodenja,
- funkcije delnega avtomatskega vodenja ter funcije osebja (operaterjev, tehničnega vodstva, poslovnega vodstva) in
- funkcije osebja.



Slika 4 Struktura tehnološkega podprocesa prikazana z diagramom toka podatkov



Slika 5 Vzorec upravljanja tehnološkega podprocesa

Proces upravljanja s podatki tehnološkega podprocesa *upravljanje* je metodološko predstavljen kot *podatkovni proces*. Obsega transformacijo vhodnih podatkov v izhodne ter določa pogoje za izvedbo določenih akcij.

Proces vodenja tehnološkega podprocesa je metodološko predstavljen kot *kontrolni proces*. Obsega transformacijo pogojev v akcije, ki imajo za posledico spremembo stanja tehnološkega podprocesa. Proces vodenja je podrobneje razčlenjen z *diagramom prehajanja stanj* (STD).

Stanje tehnološke razvitosti industrije pri nas je v povprečju tako, da obstaja avtomatsko vodenje le na najnižjem nivoju, to je na nivoju osnovne regulacije, v nekaterih kritičnih tehnoloških podprocesih. V informacijskem modelu obstoječega stanja zato nastopa precej podatkov, ki so izrazito orientirani na osnovno regulacijo tehnoloških podprocesov. Ti so razvrščeni v sledeče skupine:

- parametri tehnološkega podprocesa,
- nastavitve tehnoloških veličin in
- merjene veličine tehnološkega podprocesa.

Na diagramih tokov podatkov se pojavlja vzorec, ki prikazauje tokove teh skupin podatkov med tehnološkimi podprocesi in procesom upravljanja (Slika 5). Procesi z imenom upravljanje predstavljajo komunikacijo *procesorja* (računalnika in/ali operaterja) s tehnološkim podprocesom. Procesor sprejema navodila (*parametre tehnološkega podprocesa*) in na njihovi osnovi nastavlja ustrezne vrednosti dosegljivih procesnih veličin (*nastavitve veličin tehnološkega podprocesa*). S tehnološkega podprocesa dobi povratne informacije o dejanskih vrednostih procesnih veličin (*merjene veličine tehnološkega podprocesa*). Procesor torej ukrepa na podlagi navodil in dejanskega stanja tehnološkega podprocesa.

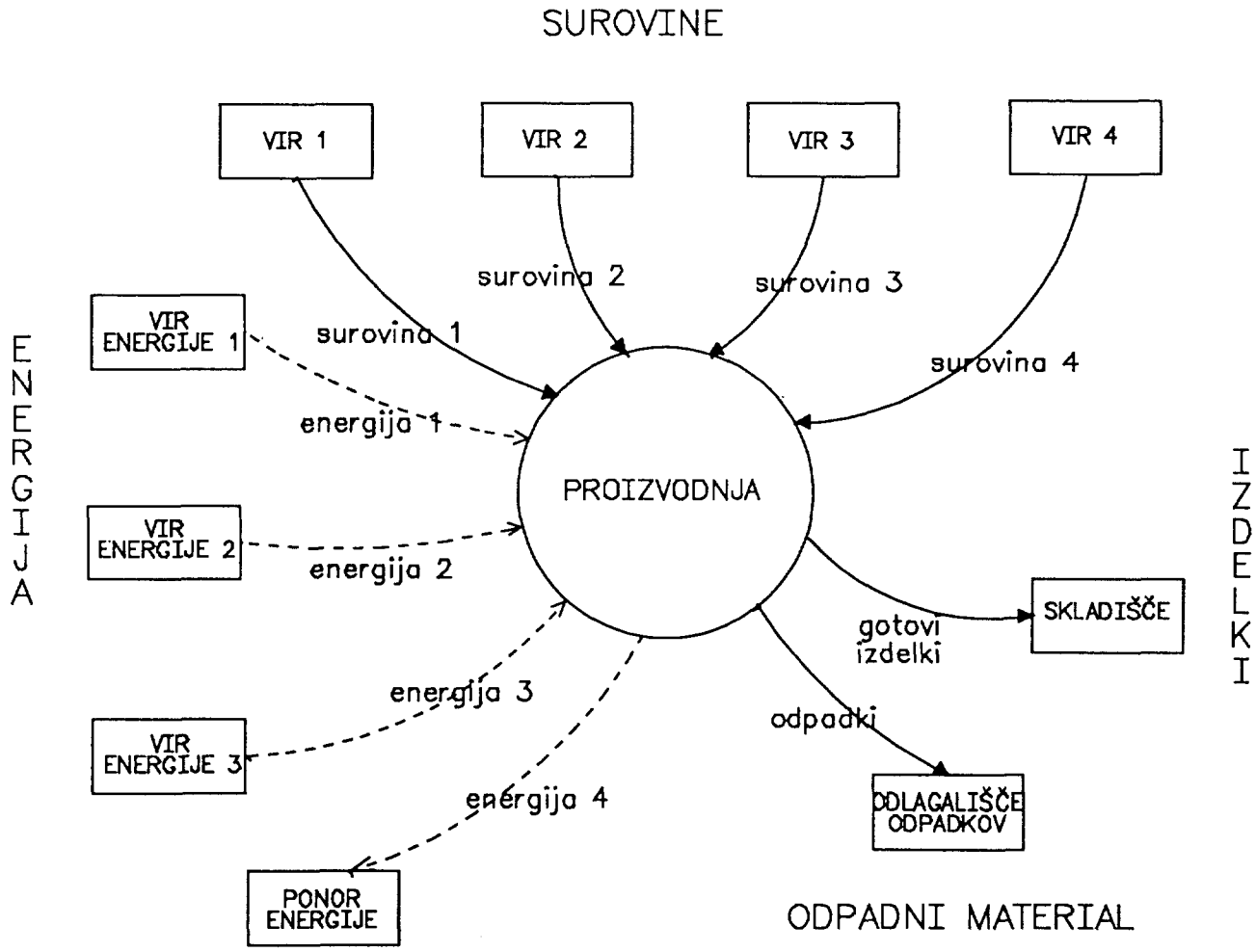
Enak vzorec skupin podatkov se pojavi tudi v *slovarju informacijskih tokov*, kjer so opisani vsi elementi, ki nastopajo na diagramih tokov podatkov in na diagramih prehajanja stanj. S postopki iskanja in sortiranja lahko iz slovarja dobimo za idejni projekt potrebne podatke, kot so npr. seznam električnih signalov, seznam merilne in regulacijske opreme itd.

4.2 Materialno-energijski model

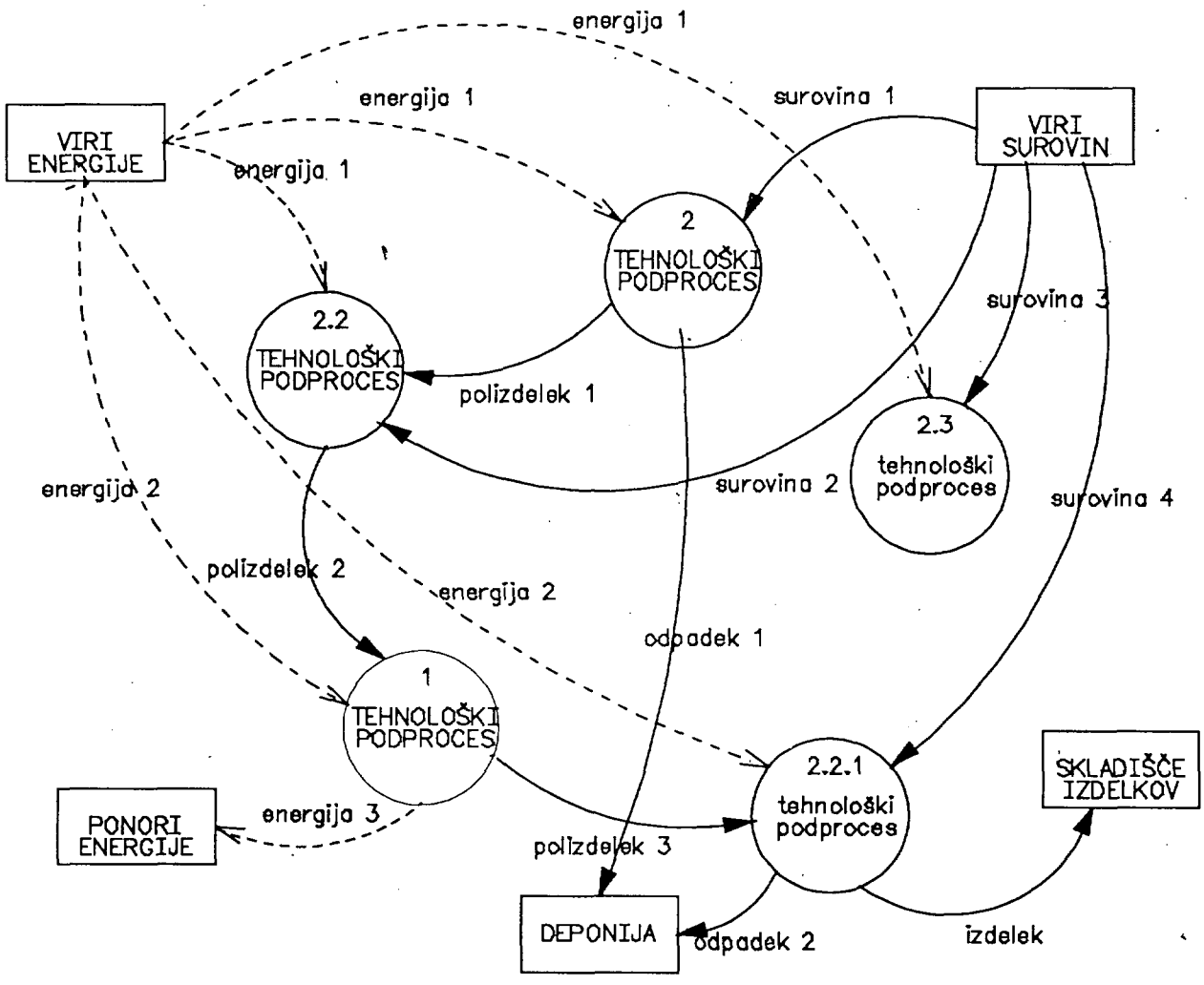
Materialno-energijski model prikazuje tokove materiala in energije med tehnološkimi podprocesi in njihovo okolico. Vsebuje vse bistvene podatke, ki so potrebni za izračun oziroma ocenitev ekonomske učinkovitosti racionalizacije porabe surovin in energije ter povečane kvalitete izdelkov kot posledico uvedbe računalniškega vodenja. Model sestavljajo *kontekstni diagram, sistemski diagram materialnih in energijskih tokov* ter *slovar.*

Kontekstni diagram materialno-energijskih tokov prikazuje (Slika 6):

- celoten tehnološki proces organizacijske enote proizvodnja kot *kontekstni proces,*
- vire materiala in energije, ponore izdelkov in odpadkov kot *terminatorje,*
- tokove surovin, izdelkov, odpadkov, energije med kontekstnim procesom in terminatorji.



Slika 6 Kontekstni diagram materialnih in energijskih tokov



Slika 7 Materialni in energijski tokovi med procesi in terminatorji na sistemskemu diagramu

Terminatorji so prostorsko grupirani. Iz terminatorjev-virov izhajajo materialni in energijski tokovi in vstopajo v kontekstni proces, iz njega pa izstopajo materialni in energijski tokovi in vstopajo v terminatorje-ponore. Energijski tokovi so označeni s črtkanimi usmerjenimi loki, pri čemer so puščice odprte. Materialni tokovi so označeni s polnimi usmerjenimi loki in počrnjenimi puščicami.

Celoten kontekstni proces je podrobneje razčlenjen na sistemskem diagramu tokov materiala in energije (Slika 7). Na njem so prikazani vsi tehnološki podprocesi v organizacijski enoti proizvodnja, ki kakorkoli preoblikujejo vhodne materialne in energijske tokove v izhodne. Za vsak tehnološki podproces je prikazana transformacija vhodnih surovin v izhodne polizdelke ali izdelke ob dovedeni energiji.

V slovarju materialno-energijskih tokov so opisani vsi elementi, ki nastopajo na kontekstnem in sistemskem diagramu. Pri materialno-energijskih tokovih je težišče opisa predvsem na ekonomskih kategorijah (cenah, porabi,...).

4.3 Povezanost informacijskega in materialno-energijskega modela

Za sistemsko predstavitev obstoječega stanja industrijskega procesa je potrebno najti način predstavitve obeh modelov, s katerim se da ugotoviti njuno medsebojno povezanost.

Pragmatično smo to rešili s poimenovanjem procesov na obeh modelih z enakimi imeni. Pri tem se moramo zavedati, da procesi v materialno-energijskem modelu transformirajo vhodne materialne in energijske tokove v izhodne s stališča snovne preobrazbe ob dovedeni energiji, medtem ko procesi v informacijskem modelu transformirajo informacije. Informacijski model je v idejnem projektu potrebno mnogo bolj strukturirati in izdelati v globino kot materialno-energijski model. *Povezanost med modeloma tako lahko prikažemo le med sistemskim diagramom materialnih in energijskih tokov ter procesi, razmeščenimi po različnih nivojih predstavitve v informacijskem modelu.* Številke procesov na sistemskem diagramu materialnih in energijskih tokov (Slika 7) se tako ujemajo s številkami procesov, ki nastopajo na različnih nivojih informacijskega modela. Tak način povezave ob upoštevanju, da imajo tehnološki podprocesi iz obeh modelov skupne specifikacije, pripomore k celovitemu pogledu na obstoječe stanje tehnološkega procesa.

Velikega pomena za naročnika je tudi povezava modela obstoječega stanja z že obstoječo tehnično dokumentacijo. Tu je možnih je več vrst povezav in sicer:

- preko slovarja podatkov, kjer je v opisu elementa (toka podatkov, materiala, energije) podana referenca na ustrezen element na obstoječi tehnološki shemi,
- preko enakih imen tehnoloških podprocesov, podatkov, itd. na obstoječih shemah in v modelu,
- preko številk procesov v modelu, ki se lahko skladajo z vodilnimi številkami oznak za posamezne tehnološke podgrupe na tehnoloških shemah, itd.

ZAKLJUČEK

Z izdelano analizo obstoječega stanja je zaključena začetna faza idejnega projekta. S preudarno izbiro razpoložljivih računalniških orodij in primerno metodološko podporo je zaključena v okviru priporočljive porabe resursov. Dokumenti analize obstoječega stanja lahko služijo kot dobra in uporabna podlaga za snovanje idejnih rešitev, nemalokrat pa so v pomoč tudi v fazi analize zahtev v bodočem projektu. Stopnja uporabnosti zavisi od izbranih rešitev, od izbrane predstavitve idej in seveda od metodološkega pristopa. V tem članku opisan pristop smo uspešno uporabili pri izdelavi Idejnega projekta za informatizacijo in avtomatizacijo izdelave mineralnih plošč.

LITERATURA

(Bowl81) Bowler T. D., General System Thinking - Its Scope and Applicability, North Holland, Oxford, 1981,

(Čern90) Černetič J, G. Godena, N. Hvala, M. Rihar: Računalniška avtomatizacija kuhanja celuloze za podjetje "VIDEM" : Osnutki dokumentov za definiranje sistema v 4. etapi projekta, IJS delovno poročilo DP-5818, 1990,

(Čern91) Černetič J., Vloga idejnega projekta pri izvajanju računalniške avtomatizacije, Zbornik referatov s posvetovanja Vodenje kemijskih procesov, Maribor, Oktober 1991, 36-45,

(Rih91a) Rihar M., Izkušnje s CASE pri strukturni analizi in specificiranju sistemov računalniške avtomatizacije, CASE III - 3. jug. sav. o metodama i alatima za projektiranje informacijskih sistema, 14.1-14.13, Opatija, 1991,

(Rih91b) Rihar M., D. Srpan, J. Petrovčič, V. Jovan, J. Černetič, S. Strmčnik, S. Pevec, Đ. Juričić: Informatizacija in avtomatizacija proizvodnje mineralnih plošč: idejni projekt, IJS delovno poročilo DP-6204, 1991,

(Šube91) Šubelj M., Pomen sistemske analize pri avtomatizaciji tehnoloških procesov, Zbornik referatov s posvetovanja Vodenje kemijskih procesov, Maribor, Oktober 1991, 28-35,

(Your89) Yourdon E., Modern Structured Analysis, Prentice-Hall, Englewood Cliffs, 1989.

Knjižna recenzija

Ivan Meško

Optimizacija poslovanja s programi na disketi

Knjiga, ki je hkrati tudi učbenik, predstavlja pomembno delo na področju optimizacije poslovanja. Je rezultat avtorjevih dolgoletnih izkušenj pri pedagoškem delu na diplomskem in podiplomskem študiju na Ekonomsko-poslovni fakulteti v Mariboru, predvsem pa njegovega uspešnega raziskovanja na fundamentalnem in aplikativnem področju.

Optimizacijske metode, ki so v razvitem svetu že nekaj desetletij nepogrešljivo orodje pri poslovnih odločitvah, se pri nas le s težavo uveljavljajo. Glavni razlogi za takšne razmere so pomanjkanje znanja pri ljudeh, ki o tem odločajo, pomanjkanje ustreznih kadrov, pomanjkljivi podatki, ki jih potrebujemo pri implementaciji kvantitativnih metod, in tudi nepredvidljive spremembe ekonomske politike.

Knjiga I. Meška je zato zelo dobrodošla, saj nudi prikaz v podjetniški praksi uspešno uporabljenih računalniških programov pri optimizaciji poslovanja.

Razdeljena je na šest poglavij, ki smiselno prehajajo iz enega v drugega.

Prvo poglavje je namenjeno splošni problematiki modeliranja poslovnih procesov. Vredno je opozoriti na točko 1.6., v kateri podaja avtor razmišljanja o pogojih za uspešno uvajanje optimizacijskih metod v poslovno prakso.

V drugem poglavju so dane osnove linearne optimizacije. Prikaz metode simpleksov je nazoren, saj se avtor izogne v tej knjigi nepotrebnemu strogemu matematičnemu dokazovanju, ki bralca običajno ne pritegne. Dan je podroben prikaz programa LOMP, s katerim uspešno rešujemo srednje velike modele.

Tretje poglavje obravnava optimizacijo enofaznih proizvodnih procesov. Predstavljen je osnovni model in podanih je več praktičnih primerov (npr.: optimizacija rezanja, optimizacija mešanja, minimizacija stroškov ob upoštevanju zalog, optimizacija transporta).

Četrto poglavje je namenjeno optimizaciji večfaznih poslovnih procesov; opisan je program LOMPV. Kot primer takšnega procesa je naveden primer iz mlečne industrije. Poseben poudarek je dan analizi oportunitetnih stroškov, ki jih avtorji običajno ne vključujejo v obravnavo.

V petem poglavju so dane osnove konveksne minimizacije. Za računanje optimalne rešitve je uporabljena metoda možnih smeri;

obrazložen je program KONO. Šesto poglavje pa je namenjeno negladkim in nekonveksnim optimizacijskim modelom. Pri teh modelih igrajo osrednjo vlogo odsekoma linearne funkcije. S takšnimi funkcijami je namreč možno aproksimirarti nelinearne funkcije. Za iskanje optimalnih rešitev omenjenih modelov sta opisana dva programa: MIXIN in LINDO.

Knjigi je priložena disketa s programi in testnimi primeri. Programi so napisani v fortranu. Podana so tudi natančnejša navodila za uporabo diskete.

Pomen knjige I. Meška moremo strniti v naslednjem:

- delo prinaša avtorjeve originalne prispevke na teoretičnem in praktičnem področju,
- delo pomeni novost v slovenskem prostoru,
- odlikuje se po razumljivi razlagi,
- »matematični jezik« je prisoten le do te mere, da bi bralec videl pot, po kateri je zgrajen določen algoritem,
- številni konkretni primeri kažejo na uporabnost modelov in tem modelom pripadajočih računalniških programov.

Ob zaključku naj navedem, da služi knjiga kot študijska literatura na dodiplomskem študiju Ekonomsko-poslovne fakultete v Mariboru. Primerna je tudi za praktike, ki želijo racionalizirati poslovne procese s sodobnimi, t.j. optimizacijskimi metodami.

Prof. dr. Stane Indihar
Ekonomsko-poslovna fakulteta
Maribor

Hubert L. Dreyfus
Being-in-the-World
A Commentary on Heidegger's Being and Time, *Division I*
The MIT Press (1991), Cambridge, Massachusetts
London, England

Lahko bi padel očitek, kaj ima delo ameriškega filozofa opraviti v informatiki in še posebej v računalništvu ali celo v umetni inteligenci. Očitek tudi ne bi bil presenetljiv, če se upošteva usmerjenost domače znanosti v klasični modernizem, le v trdo in tradicijsko (arhaično) trdno varianto znanstvenega v računalništvu; čeprav vsa pojavnost in resnica že nekaj desetletij ni več zožena na táko videnje umetne inteligence. Kar nekaj je mejnikov v razvoju kritike umetne inteligence: Dreyfus niti ni njen izvirni začetnik in Terry Winograd njen najbolj izvedenski nadaljevalec (pedagog). Začetki bržkone niso niti v kritiki Descartesovega kognitivizma in Husserlovega fenomenalizma, tj. v tradicionalni ontologiji in epistemologiji, ki sta predmet kritike v Heideggrovi filozofiji; dvom o tem je prisoten že v začetkih današnje kulture — v grštvu.

Z delom *Bit-v-svetu* postaja Dreyfus najbolj neposreden interpret Heideggrovega epohalnega dela *Bit in čas*, in sicer le prvega, po Dreyfusu bistvenega dela tega dela. Citiranost Heideggra v tej knjigi je tolikšna, da je moral avtor dobiti posebno dovoljenje za objavo od nosilcev avtorskih pravic Heideggrovih del. V predgovoru je zapisanih kar nekaj zanimivosti za občutljivega bralca. Dreyfusova knjiga je nastajala kot predavateljski zapis kurza o *Biti in času* od leta 1968 naprej. Ta zapis so popravljali avtor, njegovi asistenti in študentje. V letu 1988 je že bila pripravljena knjižna oblika dela, ki je dobila svojo končno obliko po kurzu, ki ga je imel avtor o *Biti in času* na univerzi v Frankfurtu. Avtor je tako preizkusil svoje delo s predavanji v osrčju nemške filozofije.

Delo *Sei und Zeit* (1927) je postalo klasika takoj po izidu. Kot pravi Dreyfus, ga Heidegger bržkone tudi zaradi tega ni več spreminjal. To delo je hkrati notorično trd oreh za prevajanje (v slovenščini ga še dane nimamo: prevod v japonščino l. 1939, v angleščino 1962, v hrvaščino 1985). Težava je v posebni terminologiji, ki jo je Heidegger moral vpeljati zaradi popačenosti in zmotnosti tradicionalnega jezika v filozofiji. Angleški prevod Macquarrieja in Robinsona morda danes res ni več primeren, zato predlaga Dreyfus (skupaj z J. Haugelandom in W. Blattnerjem) spremembo nekaterih terminov, ki so zanimivi tudi za slovensko pojmovanje nemških terminov: namesto directionality (Ausrichtung) *orientation (usmerjenost);* state of mind (Befindlichkeit) *affectedness (počutnost);* to encounter (begegnen) *to show up* (for us) *(srečati);* transparent (durchsichtig) *clear, perspicious (prozoren);* de-severence (Ent-fernung) *dis-tance (oddaljenost);* totality (Ganzheit) *whole (celost);* with-in-the-world (innerweltlich) *intrawordly (znotrajsveten);* the They (das Man) *the one (Se);* discourse (Rede) *telling (govorjenje);* Being *being (bit);* being-alongside (Sein-bei) *being-amidst (pri-bit);* an entity (ein Seiendes) *a being, an entity (bivajoče, entiteta);* potentiality-for-being (Seinkönnen) *ability-to-be (moči-biti);* meaning (Sinn) *sense (smisel);* uncanny (unheimlich) *unsettled (nedomač, nereden);* primordial (ursprünglich) *originary (izviren);* constitution (Verfassung) *makeup (sestava, narejenost);* anticipating (Vorlaufen) *forerun (pred-tek);* worldhood (Weltlichkeit) *worldliness (svetnost);* the upon-which (Woraufhin) *on the basis of which (načemer-se); by and large (zunächst und zumeist) primarily and usually (najprej in najbolj);* context (Zusammenhang) *nexus (povezanost).*

Knjiga ima 15 poglavij, in sicer: Heideggrov bistveni uvod, Heideggrov metodološki uvod, predhodni oris biti-v-svetu, razpoložljivost in dogodje, svetnost, Heideggrova kritika sodobnega kartezijanizma, prostornost in prostor, »Kdo« v tubiti vsakdanjosti, trojna struktura v-biti, počutnost, razumevanje, govorjenje in smisel, propadanje [zapadlost], skrb-struktura in naposled filozofske implikacije hermenevtike vsakdanjosti. Dodatek, ki ga je napisal Dreyfus skupaj z Jane Rubin, ima naslov »Kierkegaard, drugi del in pozni Heidegger«.

To, kar je gledano površno, lahko pomembno za umetno inteligenco, so poglavja o počutnosti (state of mind), razumevanju, govorjenju in smislu ter propadanju (časovnost razprtja, kjer je razprtje oblika razumevanja). Kratek pregled teh poglavij pokaže, da se Dreyfus osredotoča na posebne vidike Heideggrove filozofije, ki se mu kažejo kot bistveni. V okviru

počutnosti izpostavlja npr. razpoloženje (mood) in v okviru tega javnost (in ne zasebnost) počutja (feeling), odkrivanje (razpiranje) vrženosti in razpoloženje kot izvirno transcendenco; čustveno intencionalnost kot ontično transcendenco, in sicer strah in zaskrbljenost. V okviru *razumevanja* poskuša sistematizirati tele vidike: tri ravnine razumevanja (trenutno spopadanje kot nujnost v možnostih; igralni prostor: območje možnosti, ki so primerne v trenutnem svetu; pomembnost, svetnost in ozadje razumevanja biti); avtentično in neavtentično razumevanje (avtentično: neresnično, z bitjo zgubljeno v sebi, , resnično; neavtentično: neresnično, resnično); trije tipi razumevanja: spopadanje, interpretiranje, trdenje (izpeljava interpretacije; trojna struktura interpretacije; interpretacijske ravnine: vsakodnevno spopadanje z razpoložljivim, teorija dogodja; interpretativno razumevanje v nasprotju s teorijskim pojasnjevanjem; »trditev«: izpeljevalni način interpretacije kot kazanje na kaj ali kam, kot predikacija in komunikacija; apofantična trditev: pomanjkljivi način hermenevtične »trditve«). V okviru *govorjenja in smisla* obravnava: govorjenje kot eksistencial, jezik (jezik in smisel; komunikacija); smisel: ozadje razumljivosti; povzetek vidikov razumljivosti. *Propadanje* sodi med najbolj zapletene in nejasne koncepte Heideggrove filozofije; v okviru tega poglavja imamo propadanje kot eksistencialna struktura (absorpcija in biti zgubljeno; jezik kot izkoreninjenje; refleksivnost in popačenost); zapadlost (fallenness).

Za znanstvenika nasploh in za umetnega inteligenčnika še posebej je lahko zanimivo šesto poglavje, ki obravnava Heideggrovo kritiko sodobnega kartezijanizma. Tu najdemo: ontološki status narave (narava kot razpoložljivo: naravni materijali, naravne regularnosti in narava, kot jo privzema zgodovina; narava kot nerazpoložljivo: naravne sile; narava kot dogodje; narava preprostih ljudi in romantičnih pesnikov); Heideggrova kritika znanstvenega redukcionizma; Heideggrova kritika kognitivizma; sklep. V okviru redukcionizma omenja Dreyfus tudi t.i. obrat Winograda (z določenim sarkazmom), ko pravi *Winogradova rešitev problema pomembnosti ne deluje. Zdaj Winograd spoznava »težavnost formalizacije ozadja zdravega razuma, ki določa kateri scenariji, cilji in strategije so pomembni in kako med seboj delujejo.« Winograd je tako opustil pristop z iskalno omejitvijo in »izgubil vero« v umetno inteligenco; sedaj poučuje v okviru računalniških znanosti na Stanfordu Hiedeggra.* (Glej delo Terry Winograd and Fernando Flores, *Understanding Computers and Cognition: A New Approach for Design*, Ablex, Norwood, NJ, 1987).

S kratkim pregledom nekaterih poglavij Dreyfusove knjige seveda ni izčrpan vidik zanimivosti, ki so lahko »pomembne« (signifikantne, relevantne) za področje umetne inteligence. Dreyfus seveda ne more dokazati nemogočega v umetnem (v umetni inteligenci kot posebnem delu umetnega), dokler o umetni inteligenci kot inteligenci nemogočega razpravlja. Svoj prav ima tam, kjer ga ima tudi Heidegger, tj. v specifični omejenosti tistega dela *mathesis*, ki ga danes obvladujeta matematika in tudi tehnika s kognitivizmom, redukcionizmom in scientizmom. Vendar ne gre pozabiti, da je znanost danes sicer šele na začetku, toda vendar na poti nove, razširjene *mathesis*, ki ni zgolj matematika, tradicionalna znanost in znanstveni modernizem. Dreyfusova knjiga kaže interpretativno — skozi Heideggra — v to »matetično« smer, ki je smer informacije, informiranja ali — če hočete na področju abstraktnega in simbolnega — matemacije in matematiranja, še novih in neopredeljenih področij prihodnosti.

Anton P. Železnikar

Strokovna srečanja

(Urejeno po datumih)

Jun 1-5, 1992: **FGCS '92 — International Conference on Fifth Generation Computer Systems;** The University of Tokyo, 3-1 Hongo 7-chome, Bunkyo-ku, Tokyo 113, Japan.

Jun 9-11, 1992: **6th International Conference on Scientific and Statistical Database Management;** Institute for Scientific Computing, ETH-Zentrum, CH-8092 Zurich, Switzerland.

Jun 15-18, 1992: **PERLE '92 — Parallel Architectures and Languages Europe;** Universite Paris Sud Ersay, LRI, Bat. 490, 91405 Orsay Cedex, France.

Jul 23-28, 1992: **14th International Conference on Computational Linguistics COLING-92;** Location Nantes, France; Contact: Prof. A. Zampolli, ILC, Universita di Pisa, Via della Faggiola 32, 56100 Pisa, Italy.

Aug 3-7, 1992: **10th European Conference on Artificial Intelligence;** Austrian Research Institute for AI, Schottengasse 3, A-1010 Veinna, Austria. Topika: avtomatsko sklepanje, kognitivno modeliranje, konekcionistični in PDP modeli; tehnologija in sistemi za UI; integrirani sistemi; predstavitev znanja; strojno učenje; naravni jeziki; filozofske osnove; načrtovanje in sklepanje o akcijah; principi uporabe UI; sklepanje o fizikalnih sistemih; robotika; socialne, ekonomske, pravne in umetnostne implikacije; uporabniška povezava; verifikacija, validacija in testiranje sistemov znanja; razumevanje videnja in signalov.

Aug 24-28, 1992: **13th International Congress on Cybernetics;** International Association for Cybernetics, Palais des Expositions, Place André Rijckmans, B-5000 Namur, Belgium. Topika: evolucija in metaevolucija z vidika invariant glede na transformacijske skupine; računalniki v arhitekturi; vprašanja negotovosti v odločitvenih problemih; nevromimetične raziskave; pojmovanje obdobij in evolucije; podobne strategije v možganski zmogljivosti kognitivnih in motoričnih opravil; kibernetika in človekovo vedenje; invariante kognitivne znanosti; uporaba matematike v biologiji in medicini; zakonodaja in ureditev novodobne družbe; kibernetski sistemski pristop k zgodovini; kibernetski sistemi za interpretacijo religijskih idej; evropska integracija in njene implikacije na svetovni sistem; vzgoja za 21. stoletje; nevronske raziskave: biloški modeli in praktična uporaba; informacija: kaj je njeno bistvo? naravni sistemi: novi formalizmi in ekperimentalna verifikacija; adaptivnost, učenje in puhavost; problemi osnov inteligentnega stroja; odločitvena tehnologija za izboljšanje kakovosti upravljanja; kibernetske znanosti; kognitivno modeliranje znanja in informacijska tehnologija; itd.

Aug 30 - Sep 3, 1992: **11th IAPR International Conference on Pattern Recognition;** The University of Delft, Dept. of Electroengineering, P.O.Box 5031, 2600 GA, Delft, The Netherlands.

Sep 1-4, 1992: **CONPAR 92 — VAPP V Conference on Parallel Processing;** Location Lyon, France; Contact: Prof. M. Cosnard, Ecole Normale Superieure de Lyon, LIP, 46 Alee d'Italie, 69364 Lyon Cedex, France.

Sep 2-4, 1992: **3rd International Conference on Database and Expert Systems Application — DEXA '92;** Location: Valencia, Spain; Topics: office information systems, deductive databases, design tools, heterogeneous systems, communications, information retrieval, spatial databases, databases on supercomputers, legal information systems, databases in the humanities, historical databases, medical information systems, environment information systems, parallel database systems, computer cartgraphy, multimedia databases, visual interfaces, CASE, hypertext/hypermedia, onject-oriented databases, statistical databases, data protection, museum information systems, scientific/engineering applications, distributed applications, social/governmental information systems, CIM, expert systems processing; Contact: Prof. I. Ramos, Universidad Politecnica Valencia, Dept. Sistemas Informaticos y Computacion, Apartado 22012, E-46020 Valencia, Spain.

Sep 7-11, 1992: **12th World Computer Congress;** Location: Madrid, Spain; Contact: Fed. Esp. de Sociedades de Informatica, IFIP Congress '92, Hortaleza 104-2 Izqda, 28004 Madrid, Spain.

Sep 14-17, 1992: **Euromicro 92;** Location: Paris, France; Topics: software engineering, design automation, signal processing and systems control, parallel processing, systems architecture; Contact: Telecom (Paris), Dept. Electronique, 46, Rue Barrault, 75634 Paris Cedex, France.

A.P. Železnikar

Informatica

Editor – in – Chief

ANTON P. ŽELEZNIKAR

Volaričeva 8
61111 Ljubljana
Yugoslavia

PHONE: (+38 61) 21 43 99
to the Associate Editor;
The Slovene Society Informatika:
PHONE: (+38 61) 21 44 55
TELEX: 31265 yu icc
FAX: (+38 61) 21 40 87

Associate Editor

RUDOLF MURN
Jožef Stefan Institute
Jamova c. 39
61000 Ljubljana

Editorial Board

SUAD ALAGIĆ
Faculty of Electrical Engineering
University of Sarajevo
Lukavica – Toplička bb
71000 Sarajevo

DAMJAN BOJADŽIEV
Jožef Stefan Institute
Jamova c. 39
61000 Ljubljana

JOZO DUJMOVIĆ
Faculty of Electrical Engineering
University of Belgrade
Bulevar revolucije 73
11000 Beograd

JANEZ GRAD
Faculty of Economics
University of Ljubljana
Kardeljeva ploščad 17
61000 Ljubljana

BOGOMIR HORVAT
Faculty of Engineering
University of Maribor
Smetanova ul. 17
62000 Maribor

LJUBO PIPAN
Faculty of Electrical Engineering
and Computing
University of Ljubljana
Tržaška c. 25
61000 Ljubljana

TOMAŽ PISANSKI
Department of Mathematics and
Mechanics
University of Ljubljana
Jadranska c. 19
61000 Ljubljana

OLIVER POPOV
Faculty of Natural Sciences
and Mathematics
C. M. University of Skopje
Arhimedova 5
91000 Skopje

SAŠO PREŠERN
PAREX, Institut for Computer
Engineering and Consulting
Kardeljeva 8
61000 Ljubljana

VILJEM RUPNIK
Faculty of Economics
University of Ljubljana
Kardeljeva ploščad 17
61000 Ljubljana

BRANKO SOUČEK
Faculty of Natural Sciences
and Mathematics
University of Zagreb
Marulićev trg 19
41000 Zagreb

Publishing Council

TOMAŽ BANOVEC
Zavod SR Slovenije za
statistiko
Vožarski pot 12
61000 Ljubljana

ANDREJ JERMAN – BLAŽIČ
Iskra Telematika
Tržaška c. 2
61000 Ljubljana

BOJAN KLEMENČIČ

Ljubljana
Slovenia

STANE SAKSIDA
Institute of Sociology
University of Ljubljana
Cankarjeva ul. 1
61000 Ljubljana

JERNEJ VIRANT
Faculty of Electrical Engineering
and Computing
University of Ljubljana
Tržaška c. 25
61000 Ljubljana

Informatica *is published four times a year in Winter, Spring, Summer and Autumn by the Slovene Society Informatika, Tržaška cesta 2, 61000 Ljubljana, Slovenia.*

Volume 16 Number 1 March 1992 YU ISSN 0350 – 5596

Informatica

A Journal of Computing and Informatics

CONTENTS