© Strojniški vestnik 47(2001)1,15-27
ISSN 0039-2480
UDK 004.4:004.94
Izvirni znanstveni članek (1.01)

© Journal of Mechanical Engineering 47(2001)1,15-27
ISSN 0039-2480
UDC 004.4:004.94
Original scientific paper (1.01)

Ocenjevanje zahtevnosti postopkov

An Evaluation of Process Complexity

Romana Vajde Horvat - Tatjana Welzer Družovec - Ivan Rozman - Mirko Soković

Postopke v vseh aktivnostih lahko opredelimo z zaporedjem aktivnosti, ki se izvajajo, delovnimi proizvodi, ki so potrebni za izvajanje aktivnosti, oziroma nastanejo kot rezultat aktivnosti ter z viri, ki so potrebni za izvedbo posameznih aktivnosti. Čim bolj podrobno poznamo postopek, tem več podatkov je na voljo pri ocenjevanju posameznih projektov. V prispevku je opisan model SoPCoM (Software Process Complexity Model), ki omogoča izdelavo podrobnega opisa procesa na temelju notacije Petrijevih mrež. Model postavlja tudi mehanizem za ocenjevanje zahtevnosti tako posameznih gradnikov kakor tudi postopka kot celote. Na podlagi tako izvedenih ocenitev lahko določimo relativni delež truda, ki bo v izbranem projektu potreben za izvedbo posamezne aktivnosti. Model SoPCoM je bil razvit v okviru raziskovalnega dela na Univerzi v Mariboru.

(Ključne besede: postopki, modeli postopkovni, zahtevnost postopkov, mreže Petri)

All types of processes can be described as a sequence of activities using a set of required input and output products and resources of different types. Knowing the process in detail ensures that evaluations of the required effort for projects — conducted on the basis of a described process model — are more accurate. The Software Process Complexity Model (SoPCoM) described in this paper provides a mechanism for the description of a process in Petri-nets notation. Attributes defined in the SopCoM enable an evaluation of the complexity of each process element as well as the complexity of the process as a whole. The relative complexity of each activity is a basis for an evaluation of the effort of individual projects, conducted according to the process model. The SoPCoM was developed at the University of Maribor.

(Keywords: processes, process model, process complexity, Petri nets)

0 UVOD

Proizvodni postopki, zlasti tisti, ki temeljijo na serijski proizvodnji, so v praksi že dolgo izpostavljeni zahtevam po natančnem modeliranju in določanju. Na drugi strani se srečujemo s postopki, ki omogočajo več ustvarjalnosti in svobode pri delu (npr. načrtovanje proizvodnje, načrtovanje projektov, različni nadzorni postopki, storitveni in podporni postopki v podjetju) in jih je hkrati tudi teže natančno določiti. Kljub temu zaradi različnih vzrokov (prenova postopkov, priprave podjetij na overjanje sistemov kakovosti in podobno) podjetja v vse večji meri v smiselnih mejah modelirajo in določajo tudi tovrstne postopke.

Modeli postopkov, ki nastanejo, so v prvi vrsti namenjeni kot natančno navodilo za delo. Zaposleni tako natančno poznajo delovni postopek, ki ga morajo pri svojem delu izvesti. Vendar to ni edina možna uporabna vrednost postopkovnih modelov. So namreč tudi dragocen vir podatkov za vsebino in

0 INTRODUCTION

The strict modelling and definition of production processes, especially those based on serial production, is common in practice. Other types of processes, like processes for production planning and project planning, different control processes, service processes and other support processes, are usually more creative and innovative and therefore it is more difficult to define them in detail. Nevertheless, different motives like process reengineering and quality-management-system establishment also force organizations to model and define this type of process.

Process models are most commonly used as a guideline and as instructions for employees. Through the process model the process can be visualised in detail. Process models, therefore, are the source of knowledge about the content and the sequence of activities that have to be performed

zaporedje aktivnosti, ki jih je treba izvesti, o delovnih proizvodih, ki se pri tem uporabijo in morajo pri tem nastati, ter o virih, ki so potrebni za izvedbo posameznih aktivnosti. Te podatke lahko med drugim uporabimo pri načrtovanju posameznih projektov oziroma pri ocenjevanju potrebnega truda za izvajanje posameznih nalog pri projektih. Na kakšen način?

Postopkovni model določa vse aktivnosti, delovne proizvode in vire (ljudi, opremo, prostore ipd.) v procesu. Za vsakega izmed teh elementov lahko ocenimo, kako zahteven je: kako zahteven za izdelavo je posamezen delovni proizvod, kakšna je zahtevnost za vire, ki jih potrebujemo. Število in zahtevnost delovnih proizvodov, ki so uporabljeni oz. izdelani v okviru posamezne aktivnosti, vpliva na zahtevnost te aktivnosti.

Če upoštevamo to znanje o postopku in če uporabimo ustrezno formalno podporo, lahko opravimo potrebne izračune. V nadaljevanju bomo podrobneje predstavili model SoPCoM (Software Process Complexity Model), ki tovrstne izračune omogoča.

1 PREDSTAVITEV PODROČJA MODELIRANJA POSTOPKOV

Osnovni pojmi, ki jih srečujemo pri modeliranju postopkov, so naslednji:

1. Postopek je skupina med seboj povezanih korakov/aktivnosti, ki vodijo k zadanemu cilju, in vseh elementov za njihovo izvajanje.
2. Aktivnost je najmanjša akcija v postopku, za katero navzven ne prikazujemo njene strukture. Pogosto se pojem aktivnost enači še s pojmoma naloga in postopkovni korak.
3. Viri so ljudje ali drugi "izvajalci" posameznih aktivnosti. Drugi izvajalci so predvsem različna orodja in oprema. Vire torej delimo na:
   - vloge, s katerimi so opisane odgovornosti in pravice človeških virov ter njihova usposobljenost, ki je potrebna za izvedbo določene aktivnosti v postopku;
   - programsko opremo, ki jo sestavljajo računalniški programi oz. programska orodja, ki podpirajo ali avtomatizirajo določen segment dela, ki ga je treba opraviti v okviru aktivnosti;
   - strojno opremo, ki vključuje stroje, orodja, delovne postaje, strežnike, tiskalnike in drugo strojno opremo, na kateri poteka izvajanje postopka;
   - infrastrukturo, kamor spadajo prostori, pisarniška, logistična, komunikacijska in druga oprema, ki je pomembna za razvoj proizvoda. Infrastrukturo v postopkovnem modelu modeliramo takrat, kadar so zanjo podane specifične zahteve glede dostopnosti ali specialne opreme, vključene v razvojni postopek.
4. Delovni proizvod je izdelek, ki se uporablja znotraj postopka. Delovni proizvodi so lahko tako različni dokumenti, ki nastanejo pri izvajanju aktivnosti, različni načrti kakor tudi drugi tehnični izdelki, ki

within activities; about work products (artifacts) which are used and/or generated within the process; and about resources needed to perform these activities. This kind of knowledge can also be used for project planning and the evaluation of the required effort for specific activities within the process. How can this be done?

The process model specifies all activities, resources and artifacts. For each of these elements its complexity can be evaluated. It is possible to evaluate how complex it is to develop a specific work product and how complex are the resources involved in the development process. The number and the complexity of the work products and the resources which are used within an activity influence on the complexity of the activity itself.

Considering this predisposition and using an appropriate formal support, the complexity of a process can be evaluated. The model for such an evaluation, the SoPCoM (Software Process Complexity Model), will be described later in this paper.

1 AN INTRODUCTION TO PROCESS MODELLING

To understand the model which is presented later in the paper, some basic terms need to be defined:

1. Process - a group of interrelated steps/activities, leading to a common goal; and all of the elements necessary for their execution.
2. Activity - the smallest (atomic) action of the process. Its structure is not visible to the outside. The terms task and process step are also often used to refer to the atomic actions of the process.
3. Resources - employees and facilities needed to perform the specified activity. Types of resources are:
   - Roles - human resources, together with their responsibilities and authorities.
   - Software - includes software applications and tools, supporting or automating any segment of the activity.
   - Hardware - includes workstations, servers, printers and other hardware needed within the software process.
   - Infrastructure - includes offices, office equipment as well as logistic, communication and other accessories needed for software development. Infrastructure is modeled only when special requirements (for example, accessibility, special equipment) are required.
4. Artifact - a product which is used/produced within the process. Examples of artifacts: documents created as a result of activities, plans, diagrams, and other technical products generated

nastajajo pri izvajanju določene aktivnosti in se uporabljajo kot vhodi v naslednje aktivnosti.

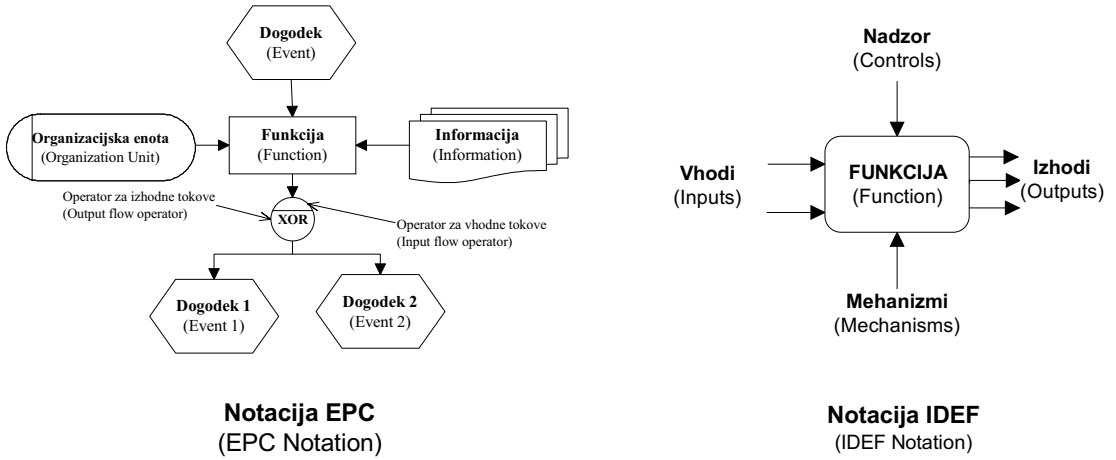
5. Postopkovni model je predstavitev določenega postopka v izbrani notaciji. V organizaciji ga lahko uporabimo kot predlogo za izvajanje dejanskih postopkov.

Naštete gradnike postopka je treba v podjetju predstaviti jasno in razumljivo. Natančnost modeliranja je odvisna od namena uporabe tako predstavljenega postopkovnega modela. Za grobo modeliranje so tako v praksi najpogosteje uporabljene metode ETVX (Entry, Task, Verification and Exit), EPC (Event Process Chain) and IDEF (Integrated Definition Language). Slednji prikazuje slika 1. Za natančnejše modeliranje postopkov uporabljamo različne metode, ki lahko temeljijo na predstavitvi stanj (avtomati stanj, Petrijeve mreže, formalne slovnice), na podlagi pravil in na podlagi ukazov ([1] in [2]).

by activities and used as inputs for the next activity.

5. Process model - a presentation of a specified process. Can be used as a template for the implementation of the real process.

All elements should be presented in a clear and understandable way. The precision of the modelling depends on the reason for using the process model. For conceptual modelling, techniques like ETVX (Entry, Task, Verification and Exit), EPC (Event Process Chain) and IDEF (Integrated Definition Language) are most often used in practice. Figure 1 presents the EPC and IDEF notations. For more detailed modelling, different methods can be used, based on state-representation-(state automates, Petri nets, formal grammars), rule- and imperative-based paradigms ([1] and [2]).



Sl. 1. *Notaciji EPC in IDEF*
Fig. 1. *EPC and IDEF notations*

2 MODELIRANJE POSTOPKOV IN VISOKONIVOJSKE OBARVANE PETRIJEVE MREŽE

Že v uvodu smo omenili, da model SoPCoM temelji na notaciji visokonivojskih obarvanih Petrijevih mrež (OPM - CPN). V nadaljevanju podajamo kratko predstavitev te notacije in primer njene uporabe za modeliranje postopkov.

2.1 Petrijeve mreže

Teorija Petrijevih mrež (PM) ima svoj začetek že v šestdesetih letih in je od takrat doživela velik razmah in vrsto razširitev. Matematična predstavitev Petrijevih mrež je dograjena s preprosto grafično predstavitvijo. Prav zaradi tega so tako pogosto uporabljene pri modeliranju postopkov. Zaradi dobre matematične zasnove so Petrijeve mreže primerno orodje tudi za analizo postopkovnih modelov. Osnovni elementi Petrijevih mrež so:

- prehodi, s katerimi predstavimo aktivnosti znotraj postopka (predstavitev: pravokotnik);

2 PROCESS MODELLING AND HIGH-LEVEL COLORED PETRI NETS

The SoPCoM (Software Complexity Process Model) is based on the notation of high-level colored Petri nets. In this section the basics of Petri-nets notation are presented.

2.1 Petri nets

Theory of Petri nets was developed in the sixties, since when it has been upgraded and enhanced in many ways. The mathematical representation of Petri nets is enhanced with a graphical representation which makes it much more understandable and more appropriate for process modelling and the further analysis of process models. The basic elements of Petri nets are:

· transitions – can be used for the presentation of activities performed within the process (presentation: rectangle);

- žetoni, ki jih uporabimo za predstavitev posameznih delovnih proizvodov in virov, ki potujejo skozi postopkovni model, (predstavitev: polni krožci);
- mesta, ki predstavljajo začasno shrambo žetonov. Žetoni na mestih čakajo, da bodo ustrezno obdelani v naslednji aktivnosti, (predstavitev: krog);
- povezave, s katerimi prikažemo mogoče poti po postopkovnem modelu (predstavitev: puščice).

Z različnimi utežmi in tipi (barvami) naštetih elementov lahko predstavimo dejansko dogajanje v postopkovnem modelu. Več o Petrijevih mrežah lahko bralec najde v našteti literaturi [2] do [5]. Slika 2 prikazuje primer postopkovnega modela, predstavljenega v notaciji Petrijevih mrež.

2.2 Primer modeliranja postopka – postopek testiranja pri razvoju programske opreme

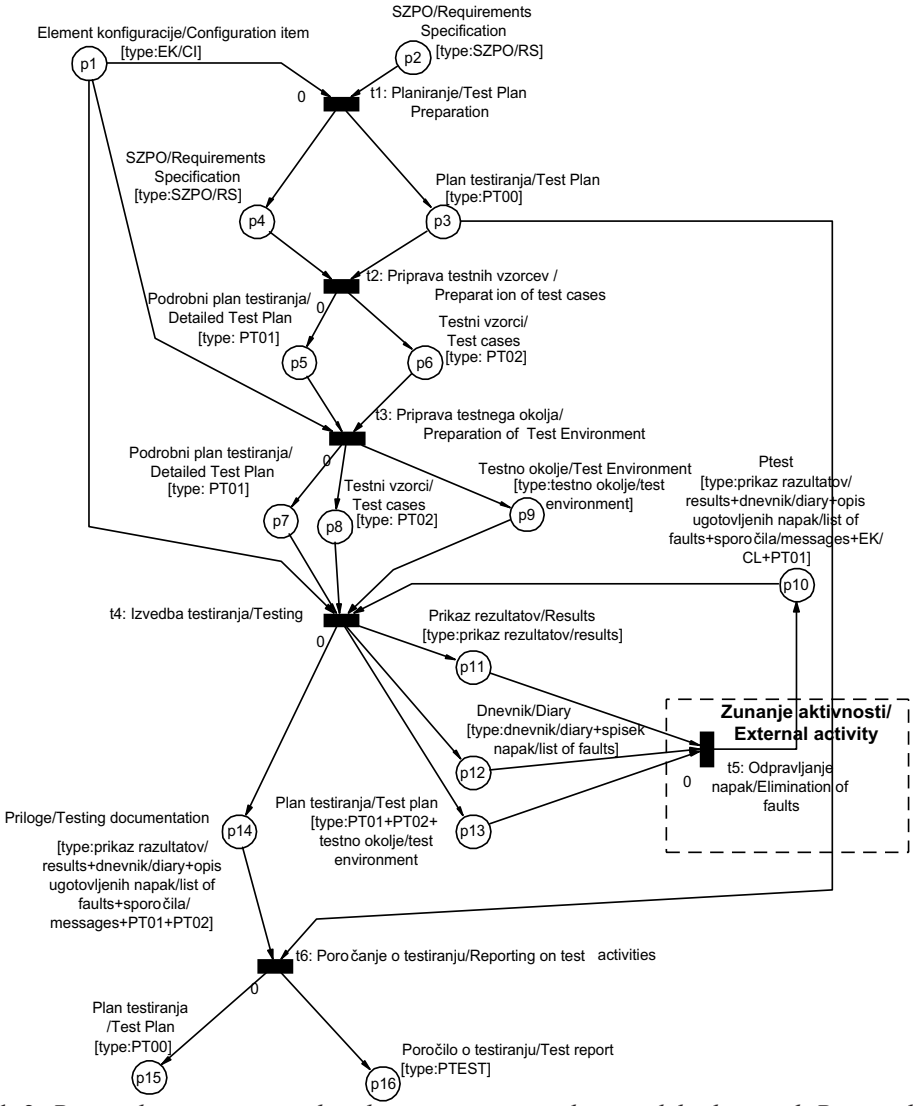
Kot tipičen primer storitvenega postopka, pri katerem se srečujemo z veliko stopnjo individualnega dela, inovativnosti in ustvarjalnosti

- tokens – can be used to represent the artifacts and resources needed to perform the activity or produced by the activity; (presentation: small tokens)
- places – represent the temporaray store for tokens. Tokens wait at places to be processed in the next activity (presentation: circle)
- arcs – represent the path of tokens between places and transitions. (presentation: arcs).

By adding different weights to these elements and by the introduction of different types (colors) of these elements, the process model can be described in detail. More about Petri nets can be found in [2] to [5]. Figure 2 shows an example of a process model represented in Petri-nets notation.

2.2 An example of a process model – Testing a process within software development

The software development process is a typical process with a high level of individualism, creativity and innovation. Consequently, it allows



Sl. 2. *Postopek testiranja, predstavljen z notacijo visokonivojskih obarvanih Petrijevih mrež*
Fig. 2. *The testing process, represented in high-level colored Petri net notation*

ter z omejenimi možnostmi predpisovanja omejitev pri izvajanju posameznih aktivnosti, lahko obravnavamo tudi postopek razvoja programske opreme. Osnovna težava pri modeliranju tovrstnih postopkov je premajhno poznavanje obsega dela, potrebnega za razvoj posameznih komponent, saj pri razvoju programske opreme dejansko načrtujemo neki proizvod (programsko opremo) in ga enkrat samkrat izdelamo. Iz tega lahko v veliki meri povlečemo vzporednice z razvojnimi postopki v proizvodnih postopkih, le da se pri programski opremi postopek po izdelavi enega proizvoda konča, medtem ko se v proizvodnih podjetjih v tej fazi običajno začne serijska proizvodnja.

Na primeru postopka testiranja pri razvoju programske opreme si poglejmo, kako je mogoče spremeniti predstavitve postopkov iz notacij EPC ali IDEF v notacijo visokonivojskih Petrijevih mrež (sl. 2).

Aktivnosti predstavimo s prehodi. Ker se lahko posamezna aktivnost izvede na več načinov (uporablja lahko različne vhode oz. ustvarja različne izhode glede na razmere v času izvajanja aktivnosti), te različne načine poimenujemo kot različne "barve aktivnosti".

Delovne proizvode, ki vstopajo v oziroma zapuščajo posamezno aktivnost, lahko modeliramo z žetoni, ki po povezavah vstopajo/zapuščajo prehod. Število teh žetonov, ki so potrebni na vhodu/izhodu krmilimo z utežmi povezav. Posamezen tip delovnega proizvoda (npr. poročilo o testiranju), predstavimo z različnimi barvami žetonov. V postopkovnem modelu dobimo torej toliko barv žetonov, kolikor je število različnih delovnih proizvodov. S tem, ko žeton določene barve postavimo na izbrano mesto, ponazorimo, da je delovni proizvod tega tipa nastal in je pripravljen za nadaljnjo obdelavo v naslednjem prehodu.

Preostale vire lahko predstavimo enako kot delovne proizvode. V nadaljevanju bomo obravnavali samo modeliranje delovnih proizvodov, ki je v praksi tudi najpogostejše, saj bo model SoPCoM tako laže razumljiv.

3 IZRAČUN ZAHTEVNOSTI MODELA POSTOPKA

Na zahtevnost modela vplivajo število in zahtevnost delovnih proizvodov in virov, ki so vključeni v postopek, ter število in zahtevnost vseh aktivnosti, ki jih v okviru postopka izvajamo.

Zgornja predpostavka zveni precej preprosto, vendar je pri izračunih treba natančno upoštevati vlogo, ki jo imajo posamezni delovni proizvodi pri posameznih aktivnostih. Izračun zahtevnosti poteka v desetih korakih, ki so predstavljeni v nadaljevanju.

3.1 Določitev vseh tipov delovnih proizvodov

Pripravimo seznam delovnih proizvodov (če modeliramo tudi vire, pripravimo tudi seznam virov). Predstavimo jih kot barve žetonov.

only limited possibilities for prescribing the boundaries for activity-implementation restrictions. The basic problem in modelling such processes is that each piece of software is actually a specific product and it is hard to predict in detail which components should be developed and what are the detailed requirements for these components. The software development process is therefore similar to the development of new products in other areas, but in the case of software the "production phase", which should succeed the development phase, does not exist.

The testing process in software development is also the example used to show the translation of EPC and IDEF presentations of the process model to Petri net notation. Figure 2 shows the example.

Activities are presented as transitions. Since each activity can be performed in different ways (using different inputs or producing different outputs depending on some conditions within the activity) these methods of activity implementation are represented as different colors of transitions.

Input/Output artifacts for an activity are modelled as tokens, which enter/leave the activity. The number of required input/output tokens is presented as the weight of a specific arc. Each type of artifact (e.g. Test Report) is presented as the color of token. In a process model we have as many tokens as we have different artifacts in a process model. A token assigned to a specific place was processed by a previous activity and is ready to be processed by the next activity (transition).

Other resources can be presented in the same way as artifacts. In practice, normally only the artifacts are modelled, and to make the SoPCoM more understandable, only the artifacts are modelled.

3 COMPUTATION OF PROCESS MODEL COMPLEXITY

The complexity of the process model depends on the number and the complexity of the artifacts and resources, as well as on the number and the complexity of the activities within the process.

It looks rather simple, but to evaluate the complexity, the roles of specific elements in the process model should be considered. The evaluation is performed in ten steps, described in the following pages.

3.1 Definition of artifacts and resources

The list of artifacts (and resources) is prepared and presented as the colors of tokens.

$\mathbf{C}_p$ = [SZPO, EK, PT00, PT01, PT02, testno okolje, prikaz rezultatov, opis ugotovljenih napak, sporočila, dnevnik, PTEST]^T

Za vrednotenje zahtevnosti delovnih proizvodov, vlog, infrastrukture, programske opreme in strojne opreme, moramo uporabiti različne atribute. Definiramo več skupin (razredov) elementov in za vsako skupino (razred) je veljaven drug nabor atributov.

Seznam razredov zapišemo kot:

$$\mathbf{G}_p = \left[g_{p1}, \ \ldots, g_{pm}, \ \ldots, \ g_{pn_{p_g}} \right]^{\mathrm{T}}$$

V primeru modeliranja več skupin je treba podati tudi pripadnost posameznih barv k določenemu razredu (na primer: barva opisa ugotovljenih napak pripada razredu delovnih proizvodov, barva vodje projekta spada v razred vloge ipd.). Seznam razredov zapišemo v vektorju $\mathbf{G}_p$, ki ima dimenzijo enako številu različnih razredov barv žetonov, ki jih pri modeliranju upoštevamo. Pripadnost posamezne barve k določeni skupini podamo s funkcijo *d* in shranimo v vektorju $\mathbf{D}_p$:

$$d : \mathbf{C}_p \rightarrow \mathbf{G}_p$$

$$\mathbf{D}_p = \left[d(c_{p1}), \ \ldots, d(c_{pn}), \ \ldots, \ d(c_{pn_{c_p}}) \right]^{\mathrm{T}}$$

3.2 Ocenjevanje zahtevnosti posameznih delovnih proizvodov in virov

Vsak delovni proizvod ocenimo na podlagi definiranih atributov, ocenjeno vrednost pa shranimo v vektor:

$$\mathbf{x}_{c_p s} = \left[x_{c_p 1s}, \ \ldots, x_{c_p ls}, \ \ldots, \ x_{c_p n_{c_p} s} \right]^{\mathrm{T}}$$

Preglednica 1 prikazuje seznam atributov za vse prej naštete razrede elementov. Za vsak atribut je določena tudi pripadajoča merska lestvica. Preglednica 2 prikazuje primer lestvice za atribut ADEF. Matematično ozadje modela SoPCoM omogoča, da si uporabnik modela sam definira razrede in atribute za ocenjevanje elementov razreda.

Ker so posamezni atributi različno pomembni za različne barve, so atributom dodeljene tudi uteži, s katerimi uravnavamo to pomembnost. Zahtevnost posamezne barve izračunamo po obrazcu:

$$x_{c_p is} = \frac{1}{n_{c_p is}} \sum_{j=1}^{n_{c_p is}} u_{c_p ij} s_{c_p ij} \qquad (1).$$

Pri tem z $x_{c_p is}$ označimo zahtevnosti *i*-te barve žetonov, z $n_{c_p is}$ število vseh atributov, s katerimi opišemo posamezni razred, $s_{c_p ij}$ so vrednosti, dodeljene posameznemu atributu in $u_{c_p ij}$ vrednosti uteži pomembnosti posameznega atributa za izbrano barvo žetonov. Zahtevnosti vseh barv žetonov (teh barv je n_{c_p}) združimo v vektorju $\mathbf{x}_{c_p s}$.

$\mathbf{C}_p$ = [SZPO, EK, PT00, PT01, PT02, test environment, results, list of faults, messages, diary, PTEST]^T

For the evaluation of the artifacts, roles, infrastructure, software (SW) and hardware (HW), different attributes should be used for each of the mentioned groups. Therefore, the artifacts and resources are grouped according to their characteristics.

The list of the groups of artifacts and resources is presented as:

When more than one group of elements is defined, the membership of elements should be defined. For example, the Test Report is the color of token, which is a member of artifact group; the project manager is a color of a token, which is a member of the roles group. The dimension of the groups vector ($\mathbf{G}_p$) is the same as the number of different groups of tokens. The membership of each element is assigned by function *d* and saved in vector $\mathbf{D}_p$:

3.2 Evaluation of the complexity of artifacts and resources

Each artifact and resource is evaluated according to an appropriate list of attributes and its complexity is saved in the vector:

Table 1 presents the list of attributes for all groups. For each attribute the pertaining measurement scale is defined. Table 2 presents the definition of the measurement scale for the ADEF attribute. The SoPCoM provides a mathematical mechanism which allows the user to identify his own groups and attributes.

Since individual attributes influence the complexity of the element differently, the weights are added to ensure an appropriate evaluation for each attribute. Complexity is expressed as:

where $x_{c_p is}$ is the complexity of *i*-th color of the tokens (i-th element), $n_{c_p is}$ is the number of all the attributes that describe the group, $s_{c_p ij}$ are the values assigned to each attribute for a specific element, and $u_{c_p ij}$ are the weights of influence for each attribute. The complexity of all n_{c_p} colors is then saved in the vector $\mathbf{x}_{c_p s}$.

Preglednica 1. *Seznam skupin in pripadajočih atributov zahtevnosti*
Table 1. *List of groups and the pertaining complexity of attributes*

SKUPINA GROUP	ATRIBUT ATTRIBUTE	OPIS DESCRIPTION
delovni proizvodi **artifacts**	**ADEF**	definiranost oblike / definition
	ARUS	ponovne uporabe / reusability
	ACON	upravljanje razporeda / configuration management
	AGEN	obveznost ustvarjanja / generating artefact req.
vloge **roles**	**REXP**	izkušnje / experience
	RSPC	specializacija / specialization
	RLEA	vodenje / leadership
	RCOM	sporočanje / communication
programska oprema **software**	**SEXP**	izkušnje / experiences
	SSUP	podpora / support
	SAVL	razpoložljivost / availability
	SEXC	izmenljivost podatkov / data exchange
	SALT	nadomestna orodja / alternative tools
strojna oprema **hardware**	**HAVL**	razpoložljivost / availability
	HPER	izvedbene karakteristike / performances
	HINT	povezljivost / interoperability
mesta **places**	**PAVL**	razpoložljivost / availability
	PTMP	temperatura / temparature
	PEQP	opremljenost / equipment
	PLOC	položaj / location

Preglednica 2. *Merska lestvica za atribut ADEF*
Table 2. *Measurement scale for the ADEF attribute*

Atribut: določenost oblike (ADEF) **Attribute: definition (ADEF)**	
0	Ni ustrezno Not relevant
0.2	Oblika je natančno določena – obstaja predloga z vnosnimi polji, ki jo izpolnimo brez modifikacij, obstajajo navodila in primeri Form defined in details – template with detailed instructions and examples exists, no modifications of template needed
0.4	Oblika je določena – obstaja predloga, vendar jo je treba dopolniti in/ali spremeniti, obstajajo navodila Form defined – template with instructions exists, needed modifications of template
0.6	Določene so točke/gradniki, ki jih je treba izpolniti/uporabiti Basic elements of artifact are defined, no template
0.8	Oblika je slabo določena – obstajajo le smernice za oblikovanje Basic guidelines for generating the artifact are defined, no template
1	Oblika ni določena – zapis v naravnem jeziku, ne obstaja predloga Form is not defined, instructions do not exist, template does not exist

3.3 Določitev aktivnosti

Določimo vse aktivnosti in vsa mesta, na katerih bodo žetoni čakali na nadaljnjo obdelavo. Zapišemo jih v vektorjih **T** in **P**. Dimenziji vektorjev sta n_t in n_p.

3.4 Določitev tipa izvajanja aktivnosti

Ker se lahko aktivnosti v procesu izvajajo na različne načine in pri tem uporabljajo/generirajo različne delovne proizvode, je treba v PM ustrezno predstaviti tudi te različne načine izvedbe

3.3 Definition of activities

All activities and places are listed in vectors **T** and **P**. The dimensions of the vectors are n_t and n_p.

3.4 Definition of types of activities

Some activities can be implemented in different ways when using/generating different artifacts. In Petri nets notation these different ways of performance for a specific activity should be modelled

posameznega prehoda. Predstavimo jih z barvami prehodov. Vsakemu prehodu dodelimo toliko barv, kolikor je število različnih možnih načinov izvedbe aktivnosti. Za vsako barvo prehoda nato določimo potrebne vhode in izhode (sl. 3).

Za primer na sliki 2 je t4 prehod , ki se lahko izvede na dva načina: en način je takšen, da v izdelku napako odkrijemo in drugi takšen, da napake ne odkrijemo. V obeh primerih so izhodi iz aktivnosti različni. Vektor barv prehodov je torej:

as well. They are modelled as colors of transitions. For each transition as many different colors are assigned as the number of different ways for performing the activity (see Figure 3).

In the example, presented in Figure 2, t4 is the transition, which can be performed in two ways: in the first case the faults in the tested element are found, and in the second case the faults are not found. In both cases different outputs are generated. The vector of colors of transitions is presented as:

$$\mathbf{C}_t = [c_{t1},\ c_{t2},\ c_{t3},\ c_{t4\_1},\ c_{t4\_2}, c_{t5},\ c_{t6}]$$

V modelu je predvideno, da lahko ima več prehodov tudi isto barvo, vendar se tak primer v praksi redko pojavlja. Za izračun pripravimo matriko tipa prehodov **C**, kjer za vsak prehod zapišemo, kakšna je verjetnost za izvedbo posameznih barv, ki so mu določene.

The model also offers the possibility to assign the same color to different transitions, but in practice this is rare. For the computation of the complexity of each activity, the possibilities for the implementation of each color should be assigned. It is expressed in matrix **C**.

$$\mathbf{C} = \begin{bmatrix} c_{11} & \cdots & c_{1n_{c_t}} \\ & \vdots & \\ c_{n_t 1} & \cdots & c_{n_t n_{c_t}} \end{bmatrix}$$

pri čemer je:

$$c_{ij} = \begin{cases} r;\ 0 < r \le 1; & \check{c}e\ \ barva\ \ pripada\ \ prehodu\ \ t_i \\ 0; & sicer \end{cases}$$

where:

$$c_{ij} = \begin{cases} r;\ 0 < r \le 1; & if\ the\ color\ belongs\ to\ \ t_i \\ 0; & otherwise \end{cases}$$

3.5 Določitev zahtevnosti posameznega načina izvedbe aktivnosti

Podobno, kakor smo določili zahtevnost barv žetonov, določimo tudi zahtevnost posamezne barve prehodov, le da so tokrat uporabljeni atributi, ki so definirani za ocenjevanje zahtevnosti barv prehodov.

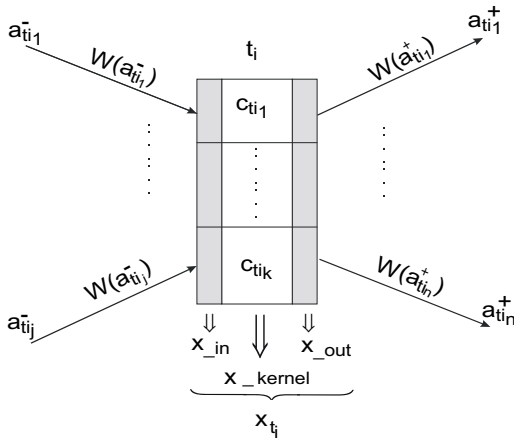
Zahtevnost vseh barv prehodov zapišemo v vektorju:

3.5 Definition of complexity for each way of activity implementation

The complexity of the color of transitions is evaluated in a similar way to the complexity of the colors of tokens. For the evaluation of the complexity of the colors of transitions specific attributes are used.

The complexities of all colors of transitions are listed in vector:

$$\mathbf{x}_{c_t s} = \left[x_{c_t 1 s},\ \ldots, x_{c_t l s},\ \ldots,\ x_{c_t n_{c_t} s} \right]^{\mathrm{T}}$$



Sl. 3. *Prikaz barve in uteženosti povezav za prehod* t_i
Fig. 3. *Schematic representation of the colors of transition* t_i

S tem, ko smo določili posamezne barve in njihovo zahtevnost, smo si pripravili vhodne podatke za izračun zahtevnosti celotnega modela. Pri tem izračunu bo treba upoštevati, kolikokrat in kako se posamezna barva žetonov uporabi pri izvajanju aktivnosti. Te podatke predstavimo z utežitvijo povezav, ki vstopajo oziroma izstopajo iz prehoda.

3.6 Določitev povezav (možnih poti pri postopku) v mreži

Vhodne in izhodne povezave za vsak prehod predstavimo v pripadajočih dveh matrikah: v matriki vhodnih povezav ($\mathbf{A}^-$) in matriki izhodnih povezav ($\mathbf{A}^+$). Poljem v matriki povezav priredimo vrednost 1, če obstaja vhodna oziroma izhodna povezava med izbranim prehodom in mestom. Matriki sta dimenzije $[n_t \times n_p]$. V praksi so vrednosti teh matrik pripravljene avtomatsko hkrati, ko povežemo posamezno mesto in prehod.

3.7 Določitev vhodnih in izhodnih pogojev za izvedbo posameznega načina aktivnosti

Število potrebnih vhodnih žetonov, ki prispejo v prehod, določimo za vsako povezavo posebej (sl. 3). Za vsako barvo prehoda moramo določiti, koliko žetonov posamezne barve mora prispeti po tej povezavi, da bo na tej povezavi izpolnjen pogoj proženja. Število teh žetonov določimo v utežni matriki povezave ($\mathbf{W}^-$ in $\mathbf{W}^+$) namreč. Takšne matrike je torej treba zapisati za vsako povezavo posebej.

Vse utežne matrike povezav združimo v ustrezni matriki utežnih matrik za vhodne povezave ($\mathbf{Y}^-$) oziroma matriki utežnih matrik za izhodne povezave ($\mathbf{Y}^+$). Dimenzije matrik so razvidne iz spodnje matrike:

$$\mathbf{Y}^- = \begin{bmatrix} \mathbf{W}^-_{11} & \cdots & \mathbf{W}^-_{1n_p} \\ & \vdots & \\ \mathbf{W}^-_{n_t1} & \cdots & \mathbf{W}^-_{n_tn_p} \end{bmatrix}$$

pri čemer je n_t število prehodov in n_p število mest.

3.8 Določitev vpliva vhodnih in izhodnih delovnih proizvodov na zahtevnost prehoda

V prejšnji točki smo določili število potrebnih vhodnih in izhodnih žetonov za proženje prehoda. Vendar ti žetoni nimajo vsi enakega vpliva na zahtevnost prehoda. Velika razlika je namreč v tem, ali na izhodu pripravimo čisto nov izdelek določenega tipa ali pa na izhod samo prenesemo nespremenjeno kopijo vhodnega izdelka. Na vhodu dodelimo vsem žetonom, ki predstavljajo delovne proizvode, utež 0,2. Drugim skupinam barv žetonov (vlogam, programski opremi, strojni opremi in prostorom) dodelimo utež

With the definition of the complexity of the colors of transitions only the basics for the computation of the whole process model were defined. What is still missing is the information about which tokens (inputs/outputs) are used/generated within each color of the transition and in what way these tokens are used. This information can be modelled by weights on the arcs entering/leaving the transition.

3.6 Definition of the arcs (paths) in the network

For each transition, input and output arcs are presented in two matrices: input-arcs matrix ($\mathbf{A}^-$) and output-arcs matrix ($\mathbf{A}^+$). The values in the matrices are 1 if the input/output arc between a specific place and transition exists, otherwise the value is 0. Matrices have the dimension $[n_t \times n_p]$. In practice, the dimensions of both matrices are generated automatically when drawing the arcs between places and transitions.

3.7 Definition of input and output constraints for each color of transition

The number of required input tokens which arrive at the transition within one arc has to be specified separately for each arc (see Figure 3). For each color of transition we should define how many tokens of a specific color should arrive in order to trigger the transition. The number of tokens is defined in arc-weight matrices $\mathbf{W}^-$ and $\mathbf{W}^+$. These two matrixes have to exist for each arc in the network.

All arc-weight matrices are joined into matrices of arc-weight matrices: for input arcs in $\mathbf{Y}^-$ and for output arcs in $\mathbf{Y}^+$. The dimensions of matrices are given below:

n_t is the number of transitions and n_p is the number of places.

3.8 Definition of the influence of input and output artifacts on the complexity of a transition

In the previous step the number of required input/output colors was defined. Each token can have an influence on the complexity of the transition in a different way. The influence is different when, for example, within the transition a new artifact is generated and when within the transition the existing artifact is only reviewed and passed to the output as an unchanged token. For all input tokens representing artifacts the influence factor 0.2 is assigned. For other groups of tokens (roles, software, hardware, infrastructure) the influence factor 1 is assigned, since

1, saj s tem ponazorimo, da uporabimo določen vir na vhodu.

Za vse vire na izhodu postavimo uteži 0 in s tem ponazorimo, da so viri po izvedbi aktivnosti spet prosti. Za delovne proizvode postavimo uteži glede na to, ali se pojavljajo kot nespremenjena kopija vhodnega delovnega proizvoda (utež 0), ali so na novo pripravljeni delovni proizvodi (utež 1) ali se pojavljajo samo kot delno spremenjeni vhodni delovni proizvodi (0<utež<1). Vrednost je v zadnjem primeru določena glede na delež spremembe vhodnega delovnega proizvoda.

Vrednosti vpliva izhodnih žetonov so določene v matriki uteži vpliva zahtevnosti barv ($\mathbf{H}^+$). Matrika $\mathbf{H}^+$ je enakih dimenzij kakor matrika utežnih matrik povezav ($\mathbf{Y}^+$).

Za izračun zahtevnosti posamezne aktivnosti je treba upoštevati vpliv vsakega delovnega proizvoda, ki vstopa oziroma izstopa iz aktivnosti.

3.9 Izračun zahtevnosti posamezne aktivnosti

Na podlagi poprej naštetih podatkov izračunamo zahtevnost posameznega prehoda po naslednji enačbi:

$$\mathbf{x}_{ts}[i] = \sum_{j=1}^{n_{c_t}} \mathbf{C}[i,j] \cdot (x\_in[i,j] + x\_out[i,j]) \cdot x\_kernel[j] \cdot 10 \quad (2),$$

pri čemer je:

$$x\_in[i,j] = \sum_{k=1}^{n_p} \mathbf{A}^-[i,k] \cdot \sum_{l=1}^{n_{c_p}} \mathbf{Y}^-[i,k,j,l] \cdot \mathbf{H}^-[\mathbf{D}_p[l]] \cdot \mathbf{x}_{c_p s}[l]$$

vsota vpliva vseh vhodnih žetonov,

$$x\_out[i,j] = \sum_{k=1}^{n_p} \mathbf{A}^+[i,k] \cdot \sum_{l=1}^{n_{c_p}} \mathbf{H}^+[i,k,j,l] \cdot \mathbf{x}_{c_p s}[l]$$

vsota vpliva vseh izhodnih žetonov in

$$x\_kernel[j] = \mathbf{x}_{c_t s}[j]$$

vpliv jedra (posamezne barve) prehoda.

3.10 Izračun zahtevnosti celotnega modela postopka

Zahtevnost celotnega modela postopka izračunamo kot vsoto zahtevnosti vseh prehodov v mreži.

$$x_s = \sum_{i=1}^{n_{tt}} \mathbf{x}_{ts}[i] \quad (3).$$

4 UPORABA IZRAČUNANIH VREDNOSTI ZAHTEVNOSTI MODELA POSTOPKA

Zahtevnost posameznih prehodov in zahtevnost celotnega modela postopka lahko pomagata pri ocenjevanju potrebnega truda za projekte,

this factor indicates that a specific resource is occupied.

For all output resources the influence factor (IF) 0 is assigned, indicating that after the completion of the transition the resource is available again. Influence factors for artifacts are assigned depending on three cases: if the output artifact is an unchanged copy of the input artifact, then IF=0; if the output artifact is newly generated then IF=1; and if the output artifact is changed from the input artifact then 0<IF<1. The value in the last case is assigned according to the percentage of artifact change.

The values are saved in the influence factor of the color complexity matrix ($\mathbf{H}^+$). Matrix $\mathbf{H}^+$ has the same dimensions as the matrix $\mathbf{Y}^+$.

To compute the complexity of each activity the influence of all input and output artifacts should be considered.

3.9 Computation of the complexity of each activity

Based on the gathered data in the previous steps the complexity of each activity can be computed according to the following equation:

where:

is the summary of the influence of all the input tokens,

is the summary of the influence of all the output tokens, and

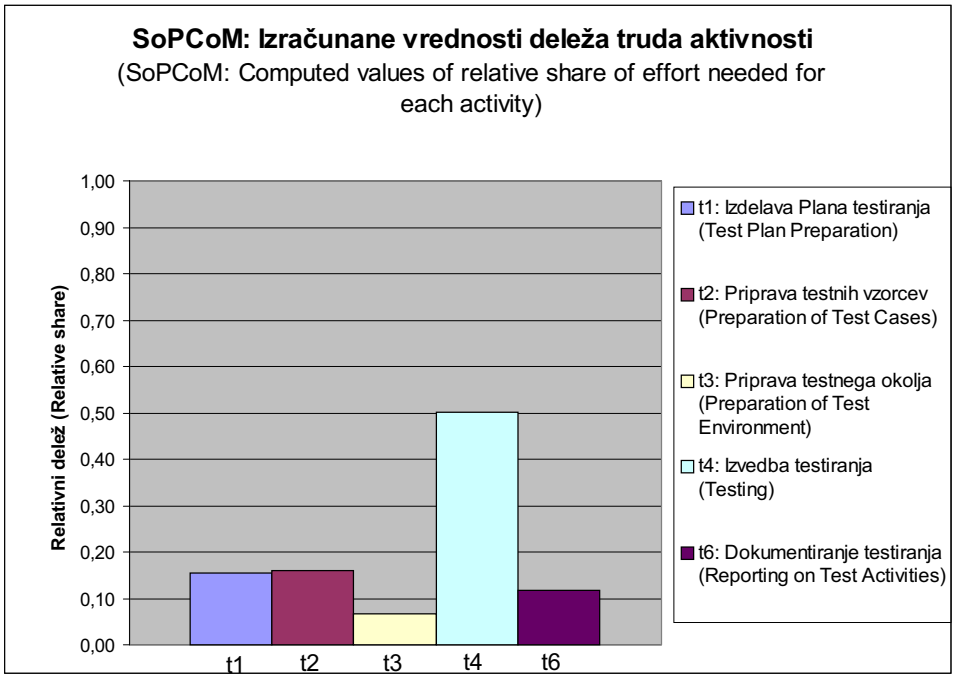
is the influence of each color of transition.

3.10 Computation of the complexity of a complete process model

The complexity of the complete process model is computed as a summary of the complexities of all the transitions in the network.

4 THE USE OF COMPUTED VALUES OF THE PROCESS-MODEL COMPLEXITY

The complexity of separate transitions and the complexity of the process model as a whole can be used for the evaluation of the effort needed for



Sl. 4. *Delež zahtevnosti posameznih aktivnosti pri postopku testiranja*
Fig. 4. *Relative share of the complexity of activities in the testing process*

ki bodo potekali po korakih, predvidenih v izbranem modelu postopka. Če namreč za posamezne prehode izračunamo njihov relativni delež v celotni zahtevnosti, dobimo podatke o deležu potrebnega truda za vsak prehod (aktivnost v projektu). Slika 4 prikazuje po modelu SoPCoM izračunane vrednosti in deleže zahtevnosti posameznih aktivnosti, predstavljenih na primeru postopka testiranja (sl. 2). Če pri izračunih zahtevnosti postopka modeliramo tudi druge vire, se absolutna zahtevnost postopkovnega modela zviša zaradi deleža, ki ga prispevajo posamezni žetoni drugih tipov na vhodu v aktivnosti (na izhodu ne vplivajo nič). Relativni delež zahtevnosti posameznih aktivnosti se v primeru, ki smo ga opisali v tem prispevku, ohranja z največjim odstopanjem 10%, kar je preverjeno tudi na drugih primerih v praksi.

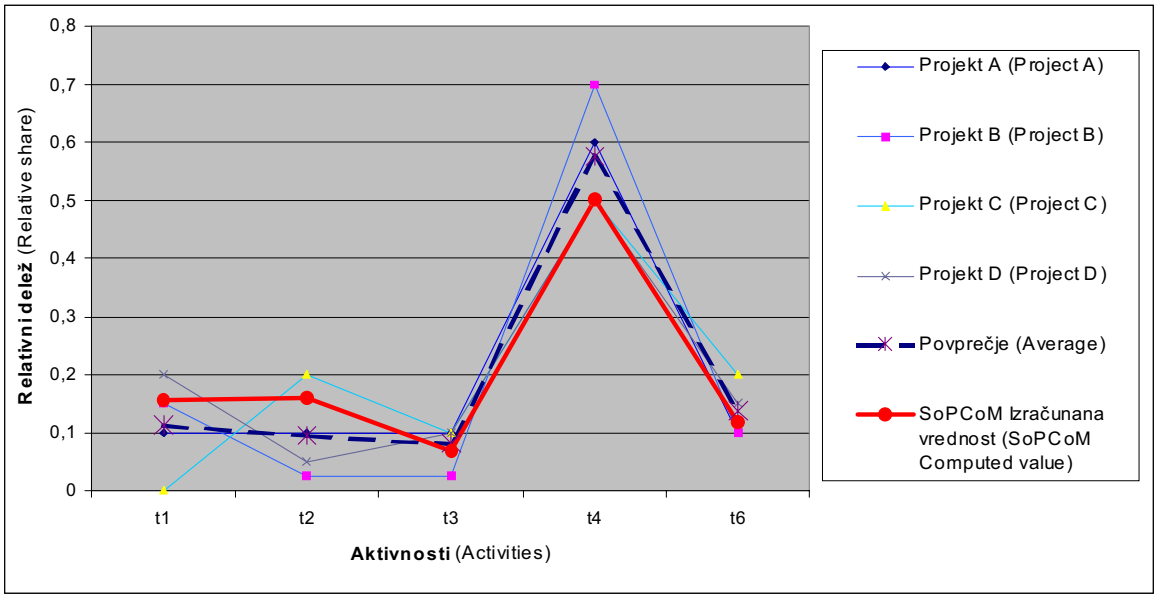
Slika 5 prikazuje rezultate preverjanja modela SoPCoM v praksi. Za štiri dejanske projekte smo pri zaposlenih izvedli modeliranje postopka ter vse potrebne ocene zahtevnosti. Ocene so podali sami zaposleni, ki so uporabo modela SoPCoM ocenili kot preprosto. Pri tem je treba upoštevati, da morajo zaposleni dejansko le pripraviti diagram poteka postopka ter ocene zahtevnosti posameznih gradnikov, ostalo delo pa opravi računalnik na osnovi ustrezne programske opreme.

Za primer Postopek testiranja smo preverili število dejansko obračunanih ur na posameznem projektu. Odstopanja dejanskih rezultatov in rezultatov izračuna po modelu SoPCoM odstopajo za največ 10%, kar je pri vnaprejšnjem ocenjevanju zelo dober približek ([7] do [9]).

projects which are performed according to the process model. If for each transition the complexity is known, the information about the required share of effort can be expressed for each transition (activity). Figure 4 shows the relative share of effort needed to implement each activity in the test process as presented in Figure 2. If other resources are also modelled the absolute value of complexity increases. The influence of input tokens on other colors (tokens representing resources) is added to the complexity of the model. Relative shares of the complexity in the example presented in this paper deviated by 10% according to shares where only artifacts were modelled. Similar results were also obtained in other cases.

Figure 5 presents the results of the practical verification of SoPCoM. For four different projects the employees provided all the required data for the description of the testing process. The employees evaluated the SoPCoM as easy to understand in those areas where they were involved. Employees only had to define the flow of the process and evaluate the complexity of each element within it according to predefined attributes. For the representation of the gathered information and further processing the software tool is used.

For the testing process the real data on used hours for the implementation of each activity in the four projects were gathered and compared to results given by the SoPCoM. SoPCoM values differ from the real data by up to 10% ([7] to [9]).



Sl. 5. *Primerjava izračunanih in realnih vrednosti za delež truda aktivnosti v postopku testiranja*
Fig. 5. *Comparison of the SoPCoM and real data on the shares of effort for activities within the testing process*

5 SKLEP

Model je bil uporabljen tudi na več drugih primerih, kjer se je izkazalo 10-odstotno odstopanje ocenjenih rezultatov od dejanskih podatkov o porabljenem času za izvajanje nalog, definiranih v obravnavanih primerih. Tovrstne rezultate je mogoče doseči le v zrelih organizacijah z zrelimi postopki. V primeru, da organizacija formalne modele svojih postopkov šele uveljavlja, lahko model SoPCoM rabi kot pomembno vodilo pri določevanju vseh potrebnih elementov procesnega modela.

Pri uporabi modela SoPCoM je treba ustrezno razrešiti tudi problem obsežnosti postopkovnega modela. Že v izbranem primeru odseka postopkovnega modela smo prišli do sorazmerno zahtevnih slik in obsežnih vhodnih podatkov. Če bi opisali postopkovni model v celoti, bi se srečali z mrežo, ki bi vsebovala veliko število prehodov, mest in povezav, prav tako bi bilo določenih veliko število barv mest in žetonov ter barv prehodov. Preglednost takšne mreže je težko zagotoviti brez podpore ustreznega orodja. Na Inštitutu za informatiko poteka razvoj orodja, ki olajša samo modeliranje in hkrati podpira algoritem modela SoPCoM.

5 CONCLUSION

The model has already been used in some other cases where a less-than 10% deviation of the SoPCoM results from the real data was also achieved. Such results can be achieved within mature organizations with mature processes. In cases where organizations are starting to establish the formal process model the SoPCoM could be used as a guideline for defining all the required elements within the process.

Further more, when using the SoPCoM, appropriate consideration should also be given to the extensiveness of the process model. Already for the simple example presented in this paper a complex figure presenting the process model was used, showing a relatively large number of artifacts. If the whole process of software development (not only the testing phase) were to be described, the process model would be very extensive. To deal with this extensiveness, an appropriate tool which supports the SoPCoM should be used. The SoPCoM Tool which is being developed at the Institute of informatics will support the modelling as well as the implementation of a SoPCoM algorithm.

6 LITERATURA
6 REFERENCES

[1] Finkelstein, A., J. Kramer, B. Nuseibeh (1994) Software process modelling and technology. *Research Studies Press Ltd., John Willey & Sons Inc. New York.*

[2] High-level Petri Nets - Concepts, definitions and graphical notation. *Committee Draft ISO/IEC 15909,* October 2, 1997, Version 3.4.

[3] Proth, J.-M., Xie, X. (1996) Petri nets - a tool for design and management of manufacturing systems. *John Wiley & Sons, Inc. New York.*

[4] Reisig, W., Rozenberg, G. (1998) Lectures on Petri nets I: Basic models. *Springer, Berlin.*
[5] Jensen, K. (1996) Colored Petri nets: Basic concepts, analysis methods and practical use (Monographs in theoretical computer science). *Springer Verlag*.
[6] http://www.dsi.unimi/Users/Tesi/trompede/petri/nets/TPN.html
[7] Vajde Horvat, R., Rozman, I. Rozman, T. (2000) How to evaluate the complexity of software processes?, *23rd International Convention MIPRO 2000*, May 22-26, Opatija, Croatia, 29-32.
[8] Vajde Horvat, R., Rozman, I., Rozman, T. (2000) SoPCoM - model za ocenjevanje kompleksnosti programskih procesov. Dnevi slovenske informatike, Portorož, Slovenija, 19.-22. april 2000. *Zbornik posvetovanja DSI 2000*, Ljubljana: Slovensko društvo Informatika, del. 1, 121-129.
[9] Vajde Horvat, R. (2000) Kompleksnost nadzora programskih procesov : doktorska disertacija. Fakulteta za elektrotehniko, računalništvo in iformatiko, Maribor: [R. Vajde-Horvat], 159 f. + pril.

Naslova avtorjev: doc.dr. Romana Vajde Horvat
prof.dr. Tatjana Welzer Družovec
prof.dr. Ivan Rozman
Fakulteta za elektrotehniko,
računalništvo in informatiko
Univerze v Mariboru
Smetanova 17
2000 Maribor

doc.dr. Mirko Soković
Fakulteta za strojništvo
Univerze v Ljubljani
Aškerčeva 6
1000 Ljubljana

Authors' Addresses: Doc.Dr. Romana Vajde Horvat
Prof.Dr. Tatjana Welzer Družovec
Prof.Dr. Ivan Rozman
Faculty of Electrical Eng. and
Computer Science
University of Maribor
Smetanova 17
2000 Maribor, Slovenia

Doc.Dr. Mirko Soković
Faculty of Mechanical Eng.
University of Ljubljana
Aškerčeva 6
1000 Ljubljana, Slovenia

Prejeto: 19.6.2000
Received:

Sprejeto: 12.4.2001
Accepted: