Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informačních technologií

Analýza příčinných smyčkových diagramů

Bakalářská práce

Autor: Tomáš, Petržilka

Studijní obor: Informační management

Vedoucí práce: doc. Ing., Vladimír Bureš, Ph.D., MBA

Hradec Králové Srpen 2018

Prohlášení:

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a s použitím uvedené literatury.

V Praze dne 18.7.2018 Tomáš Petržilka

Poděkování:

Rád bych na tomto místě poděkoval vedoucímu bakalářské práce doc. Ing., Vladimíru Burešovi, Ph.D., MBA za vedení této práce a za čas, který mi věnoval. Dále bych rád poděkoval své rodině, která mě podporovala a motivovala k zvládnutí praktické části, na které je tato práce postavena.

Anotace

Systémová dynamika je univerzální a široce používaná vědní disciplína, která zkoumá chování systémů v průběhu času. Diagram kauzálních smyček (dále jen diagram) je jeden z nástrojů systémové dynamiky. Práce se zabývá praktickou konstrukcí programu pro mechanickou analýzu diagramu vytvořeného v programu Vensim. Program přečte datový soubor MDL (interní formát Vensim), identifikuje v souboru část s diagramem a zkontroluje jeho správnost a úplnost. Nalezené nedostatky program vypíše do protokolu a do kopie diagramu. Program umí porovnat dva diagramy na základě podobnosti struktury a v obou diagramech vyznačit společné a rozdílné prvky. Protokol z programu je možné exportovat do textového souboru. Programu je samostatně spustitelný v prostředí MS Windows a napsaný v programovacím jazyce Microsoft Visual FoxPro. Program slouží pro rychlou validaci diagramů a kontrolu správnosti jejich struktury.

Klíčová slova: příčinný smyčkový diagram, strojová analýza, souborový formát MDL, Vensim, porovnání modelů.

System dynamics is a universal and widely used science discipline that examines the behavior of systems during the time. The causal loop diagram (hereafter diagram) is one of the tools of System dynamics. Thesis deals with the coding algorithm used for mechanical analysis of diagram in the Vensim data format. Program reads the MDL data file, identifies the part of the diagram in the file and checks its content against rules and completeness. The shortcomings found are listed in the log and included in copy of the diagram. The program also compares two diagrams and, on the basis of the similarity of the structure in both diagrams, indicates common or different elements. The protocol from the program can be exported to a text file. The program is self-executable in MS Windows and written in Microsoft Visual FoxPro programming language. The program is used to quickly validate models and to check the accuracy of the structure.

Keywords: causal loop diagram, machine analysis, file format MDL, Vensim, model comparison.

Obsah

Úvod

Cílem mé práce je vytvoření programu, který přečte model vytvořený v programu Vensim a ověří správnost příčinného smyčkového diagramu uloženého v tomto modelu.

Program dále zkontroluje dva diagramy (např.od dvou tvůrců) a porovná je podle jejich obsahu. Shodné, či odlišné části diagramů program zvýrazní a uloží do nového souboru. Ten je možné otevřít poté v programu Vensim a zvýrazněné části diagramu vidět v grafické reprezentaci.

O nalezených chybách a změnách program vytvoří protokol.

Teoretická část

Pokud se blíže podíváme na příčinný smyčkový diagram, jedná se o zápis struktury ohraničeného systému. Ten je složen z vlastností a vlastnosti jsou propojeny vazbami, které vyjadřují působení jedné vlastnosti na jinou ve směru vazby.

Co umí příčinný smyčkový diagram

Při konstrukci diagramu identifikujeme hlavní proměnné a dále všechny vazby, které mohou na jednotlivé proměnné působit. Postupně diagram obohacujeme o další proměnné a další vazby. Při tomto rozšiřování musíme brát ohled na to, že vazba umí vyjádřit odkud a kam působí, ale nelze vyjádřit sílu této vazby. Abychom udrželi diagram srozumitelný, což je základním smyslem a výhodou příčinných smyčkových diagramů [Richardson 1986], je třeba vždy hledat pouze nejpodstatnější vazby a proměnné. Snadno se jinak může stát, že účastník diskuse bude nad modelem pochybovat o uvedených proměnných a vazbách. Ve snaze o představu praktického fungování modelu bude přemýšlet o svých (podstatných) proměnných. Pokud ale v diagramu budou zvoleny pochopitelné proměnné a srozumitelné vazby mezi nimi, je dobrým nástrojem k tomu, aby jednoduše popsal strukturu systému a převažující interakce v něm.

Cílem diagramu je identifikovat mezi vazbami uzavřené okruhy po sobě dokola působících vazeb, které v dlouhém čase a mnoha průchodech okruhem popisují převažující chování celého systému. I zde dbáme na to, aby byly tyto smyčky ty podstatné a nebylo pochyb o tom, že teoreticky popsané interakce nebudou ve sporu s tím, jak popisovaný děj vnímá účastník diskuse.

Díky tomu, že příčinné smyčkové diagramy neobsahují na rozdíl od diagramu toků a stavů ani úrovně (levels - vyčíslené hodnoty proměnných) ani míry změny (rates - funkce působící proměnné), musíme dbát zvýšené pozornosti zejména při souběhu

dvou smyček v jedné proměnné. V takových případech musíme volit proměnné a smyčky tak, aby jejich chování bylo natolik výrazné, že i z jejich případného souběžného působení bude zřejmé, jak se celý systém chová. To je také základní nevýhodou, která je vytýkána [Richardson 1986].

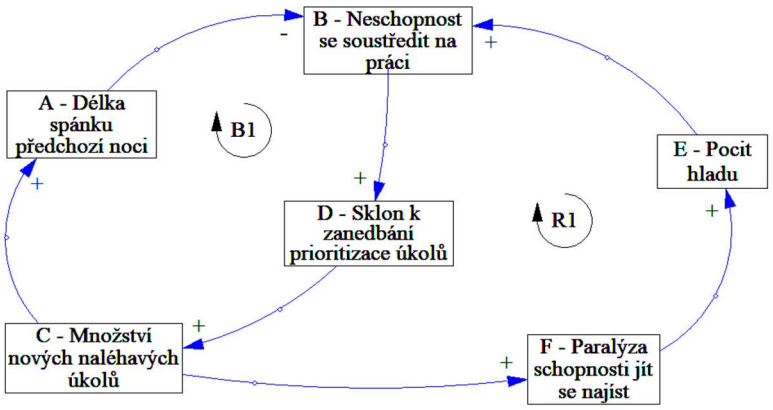
Pokud v systému identifikujeme smyčku, můžeme dále popsat chování takové smyčky. Vzhledem k tomu, že smyčka působí v kruhu a v dlouhém období předpokládáme opakovaný průchod celým kruhem, je možné, že cílová hodnota proměnné, ze které jsme vyšli, bude na konci průchodu jiná a tato výsledná hodnota bude brána jako startovní pro další průchod [Bureš 2013]. Tak je možné rozeznat posilující chování takové smyčky (používáme označení reinforcing - R) kde se proměnné mění stále stejným směrem, a nebo rozeznáváme vyrovnávající chování smyčky, kdy se proměnná v jednom průchodu mění jedním směrem a v dalším průchodu směrem opačným. Zde hovoříme o vyrovnávající (balancing – B) smyčce.

Pokud máme v diagramu více takových smyček, které jsou určitým způsobem uspořádané, můžeme v nich rozeznat některý z archetypů popsaných v [Bureš 2013] a dosáhnout přesnější představy o chování diagramu s několika smyčkami. Chování archetypů je známé a podobně očekávatelně se bude chovat i náš diagram.

Struktura diagramu

V diagramu rozeznáváme uzly, vazby a smyčky

1. uzly - proměnné, zde černě orámované obdélníky s názvem proměnné uvnitř
2. vazby - modré šipky mezi uzly
3. kauzální smyčky - zde hodnoty R/Bx v kruhové šipce

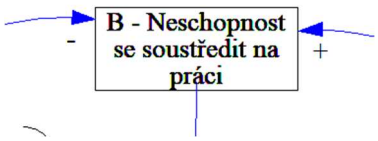


Obrázek 1 – příčinný smyčkový diagram

Uzly

Uzly jsou body v systému, reprezentované proměnnou, která nabývá určité hodnoty.

Například v obrázku 1 - "B - Neschopnost se soustředit na práci" může být nízká nebo vysoká nebo délka spánku může být krátká nebo dlouhá.



Obrázek 2 - uzel v diagramu

Proměnné v uzlech mají určitou počáteční hodnotu a díky činnosti systému se tato hodnota zvyšuje, snižuje nebo nemění a to různě a různým tempem v čase podle působíchch vazeb.

Vazby

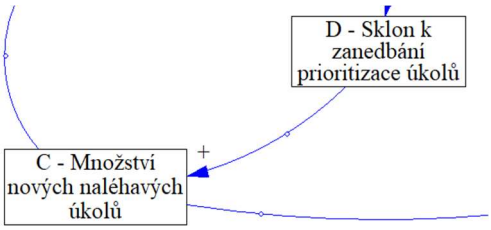
Pokud existuje mezi dvěma uzly závislost, tedy se při změně jedné proměnné mění jiná proměnná, říkáme, že je mezi oběma proměnnými vazba.

Vazba obsahuje informaci z jakého uzlu a do jakého uzel vliv působí a jestli se mění hodnota závislé proměnné stejným směrem (posilující) nebo opačným směrem (oslabující) při změně zdrojové proměnné.

Graficky vyjádřeno modrou šipkou, která vede od uzlu, který ovlivňuje, k uzlu, který je ovlivňován. Posilující vazbu značíme znaky "+" nebo "s", oslabující vazbu značíme "-" nebo "o".

Posilující vazba

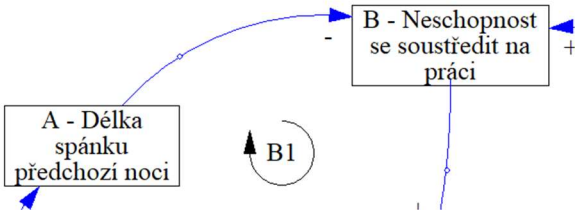
Na obrázku 3 uzel "D - Sklon k zanedbání úkolů" ovlivňuje uzel "C - Množství nových naléhavých úkolů. Z praxe jistě znáte, když příchozí frontu požadavků neprioritizujete a máte za chvíli množství stejně "naléhavých" úkolů a je pak obtížné vůbec někde začít.



Obrázek 3 - posilující vazba mezi dvěma uzly

Oslabující vazba

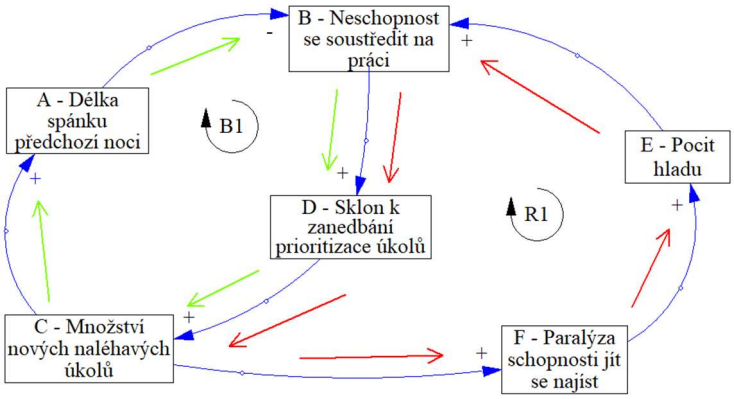
Oslabující vazba mění hodnotu cílové proměnné opačným směrem, než kterým se mění působící proměnná. Například na obrázku 4 proměnná "A - Délka spánku předchozí noci" svým růstem (déle spím) snižuje proměnnou "B - Neschopnost se soustředit na práci", protože když jsem dobře vyspaný, tak se mi na práci soustředí dobře.



Obrázek 4 - oslabující vazba mezi dvěma uzly

Kauzální smyčky

Pokud jsme v systému identifikovali všechny vazby, můžeme si všimnout, že v některých případech vazby tvoří uzavřenou smyčku. Tyto okruhy nazýváme kauzální smyčky a v diagramu vyznačují ustálené cesty, kterými dochází k ovlivňování proměnných. Na obrázku 5 vidíme 2 smyčky (zelené a červené šipky).



Obrázek 5 - Dvě kauzální smyčky v diagramu

Protože se jedná o uzavřené okruhy, budeme se zabývat otázkou, jak se bude model chovat ve druhém, třetím a dalších průchodech smyčkou, kde výsledná hodnota proměnné z prvního průchodu je vstupní hodnotou pro další průchod a tak její růst, pokles nebo setrvání může být zásadní pro chování modelu při velkém počtu průchodů.

Rozeznáváme 2 typy smyček:

1. Urychlující (R) - hodnoty proměnných nabývají neomezeně velkých (malých) hodnot
2. Vyrovnávající (B) - hodnoty proměnných konvergují k nějaké hodnotě nebo oscilují v určitém rozmezí

Urychlující (reinforcing) smyčka

Urychlující smyčky můžeme rozdělit na

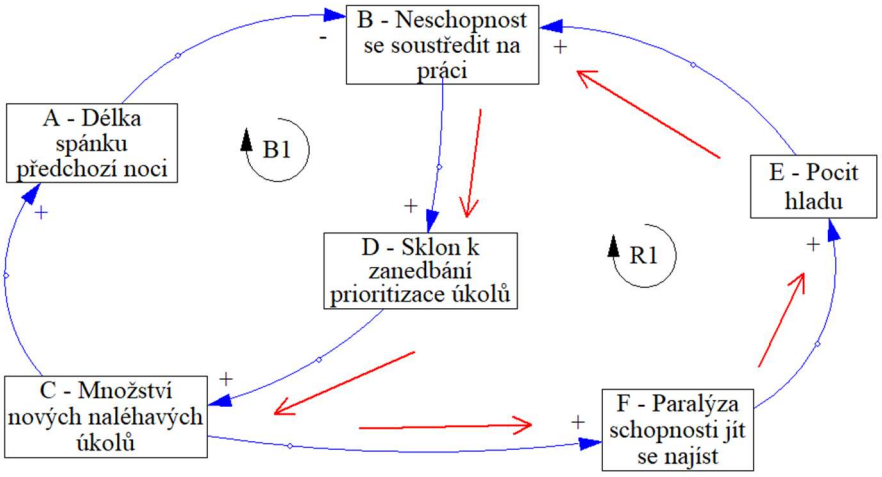
1. Smyčky skládající se pouze z posilujících vazeb
2. Smyčky složené z posilujících i oslabujících vazeb

Každý uzel, do kterého vede posilující vazba, je posílením předchozího uzlu také posílen. Řetězem posilujících vazeb mezi uzly bude systém růst a s každým dalším průchodem smyčky dál.

Uvažme scénář, kdy je jeden uzel ve smyčce příchozí vazbou oslabován. Pokud se takto "oslabený uzel" v další části smyčky propojí s dalším uzlem oslabující vazbou, je uzel za touto druhou oslabující vazbou zpátky posílen. Pokud je tedy počet oslabujících vazeb sudý, celá smyčka je také posilující.

Graficky značíme písmenem R a pořadovým číslem, zasazeným do kruhové šipky ve směru hodinových ručiček.

Příklad:



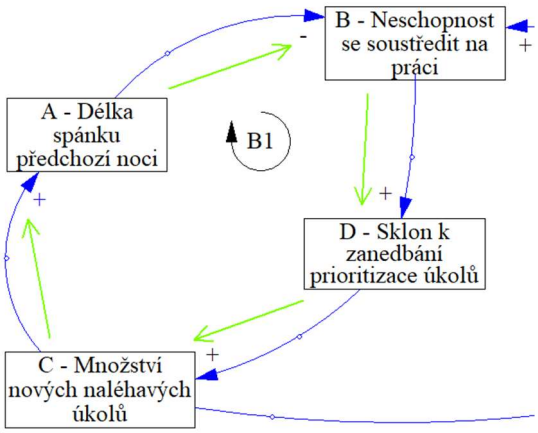
Obrázek 6 - posilující smyčka R1

Vyrovnávající (balancing) smyčka

Pokud je ve smyčce lichý počet oslabujících vazeb, dochází v průchodu smyčkou k jednomu nebo více zpomalení a vyrovnání růstu z předchozích posilujících vazeb.

Graf jednotlivých proměnných v dlouhém čase pak pak může nabývat různých tvarů, s jistotou je možné pouze říct, že oslabující vazba způsobí v dosavadním růstu nebo poklesu změnu, která může mít dle síly oslabující vazby různé podoby. To je také kritizováno jako nedostatek diagramu kauzálních smyček, kde chybí vyjádření síly vazby a nemohu tedy z jistotou říct, zda se při kumulaci několika oslabujících vazeb celkový růst ve smyčce zastaví, nebo pouze zpomalí a jak. [Richardson 1986].

Příklad rovnovážné smyčky je na obrázku 7, kde pokud přicházím do práce nevyspalý, nemohu se příliš soustředit na práci, díky tomu začnu zanedbávat prioritizování úkolů, ty se mi nahromadí a čím více jich mám, tím více mě mozek tlačí k tomu, abych si šel včas lehnout a druhý den ty úkoly roztřídil a zvládl odbavit správně.



Obrázek 7 - vyrovnávací smyčka v diagramu

Rozpoznání smyčky

Narozdíl od proměnných a vazeb, které je možné z diagramu jednoduše přečíst, je nalezení smyčky složitější. Nejsou totiž v souboru s modelem nijak propojeny s vazbami a zapsány jsou pomocí značky „komentář“. Program tak musí všechny smyčky v diagramu vyhledat.

Pokud se podíváme na definici příčinného smyčkového diagramu [Binder 2014, str. 7] hovoří o diagramu jako o orientovaném grafu. Vzhledem k tomu, že potřebujeme graf procházet, nabízí se použít matici sousednosti [Jirovský 2010] a u každé vazby určovat polaritu znaménkem u hodnoty.

Lze tak vyřešit problém vícenásobné vazby mezi dvěma body. V matici zapisujeme absolutně počet vazeb, již bez polarity (dvě vazby s opačnou polaritou by se od sebe

odečetly a nepoznáme, jestli se jedná o proměnné bez vazby, nebo s jednou + a jednou -.

Pokračujeme algoritmem pro průchod diagramem (dále jen grafem). Protože hledáme všechny smyčky, které v grafu jsou a nevíme, jestli v některých bodech nebude více smyček, metody průchodu grafem "do hloubky" a do "šířky" [Večerka 2007] pro nás nebudou vhodné. Museli bychom je spouštět pro každý uzel (kde by nám zřejmě v pořádku jedním průchodem algoritmus našel jednu smyčku), ale i tak je možné, že v uzlech, kde prochází více než jedna smyčka, algoritmus zvolí tu nesprávnou a nezískáme ani při vícenásobném spuštění všechny smyčky v grafu.

Řešení, které jsem zvolil, vychází z průchodu maticí sousednosti, kde pro každý uzel, ze kterého vede vazba, tuto vazbu vezmu a vložím do seznamu potenciálních smyček. Zároveň projdu všechny dosud vložené vazby a pokud některá z těchto vazeb začíná koncovým uzlem nebo končí počátečním uzlem zpracovávané vazby, je předchozí vazba rozšířena právě o zpracovávanou. Složitost takového algoritmu je k.n^2, kde n je počet proměnných v diagramu. Stanovení koeficientu k je vysvětleno dále.

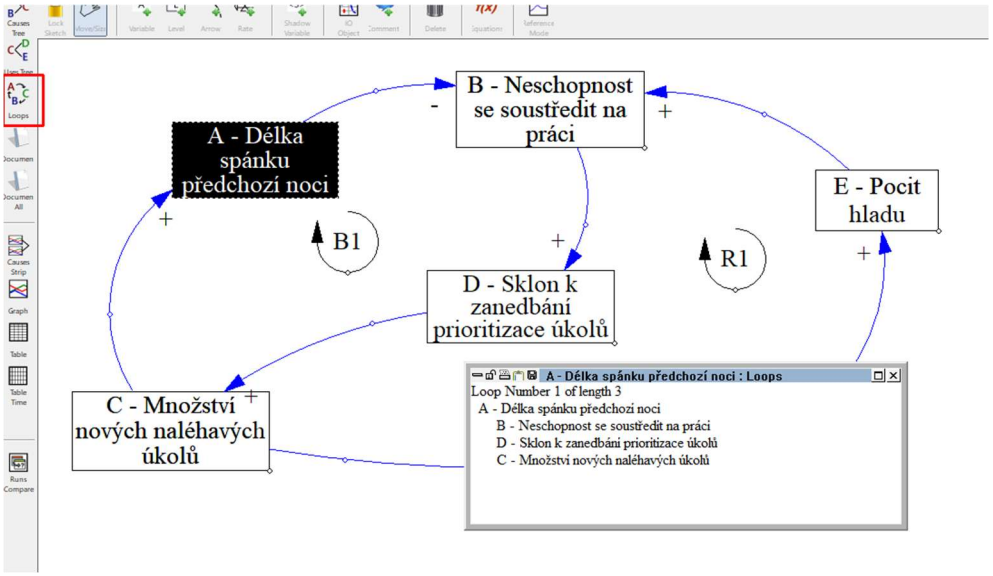
Metody analýzy diagramu

Přístupy k analýze diagramů můžeme rozdělit do dvou kategorií. V první kategorii, můžeme ji také nazvat "mechanická", nevíme, o čem se v diagramu pojednává, proměnné jsou pro nás pouze objekty, které se nějak jmenují a někde v diagramu leží, vazby pak tyto proměnné propojují a smyčky jsou pro nás komentáře s kulatou šipkou.

V kategorii druhé, nazvu ji "obsahová" již obsahu diagramu rozumíme a analýza je postavena na práci s tímto obsahem. Obsahová analýza stojí na pomyslné pyramidě potřeb o stupeň výš než mechanická (musím rozumět obsahu), ale také vyžaduje, aby byly nižší stupně, na kterých staví, v pořádku.

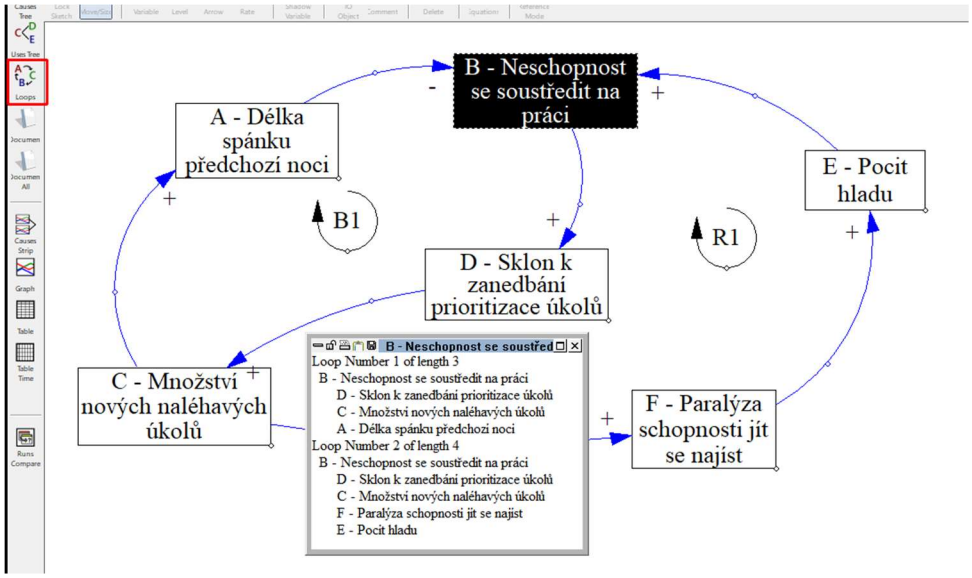
Mechanická analýza

Při tomto přístupu se zabýváme ověřením, zda diagram splňuje všechna pravidla nutná pro jeho konstrukci. Tato pravidla lze algoritmizovat a vytvořit tak nástroj na kontrolu správnosti diagramu.
Software Vensim, s jehož formátem diagramu analýzu provádíme, obsahuje interaktivní nástroje na kontrolu diagramu.
Uživatel může v modelu označit konkrétní proměnnou a spuštěním nástroje "Loops" [Vensim 2018] prověřit, jaké smyčky proměnnou procházejí:



Obrázek 8 - ukázka funkce “Loops” v programu Vensim s jednou smyčkou

V modelu vidíme dvě smyčky, pokud označím jako zkoumanou proměnnou “B - Neschopnost…”, je vidět, že proměnnou procházejí obě tyto smyčky:



Obrázek 9 - ukázka funkce “Loops” v programu Vensim se dvěma smyčkami

Tento nástroj je vhodný pro průběžnou kontrolu nebo pro hledání příčiny konkrétní chyby v diagramu při modelování.

Označení nalezených smyček R1 a B1 je provedeno pomocí prvku “komentář” a není se smyčkami v diagramu nijak provázáno. V programu Vensim tak není možné zkontrolovat, zda vyznačené smyčky odpovídají těm, co v diagramu skutečně existují.

Tento fakt je impulsem pro moji práci, ve které se s použitím mechanického přístupu věnuji kontrole již hotového diagramu a dále nalezení všech smyček algoritmicky. Zkoumám tedy, zda jsou vazby mezi proměnnými správně označeny polaritou, identifikuji v diagramu všechny smyčky a porovnávám, zda označení těchto smyček v diagramu odpovídá skutečnosti.

Při stanovení algoritmu pro hledání smyček je zajímavé inspirovat se prací [Oliva 2004], která řeší problematiku nalezení smyček v modelu. Používá také matici sousednosti pro zápis proměnných a vazeb, ale tuto dále rozvíjí o matici vzdálenosti proměnných od sebe. Nalézá smyčky postupně od malých po větší a vždy při vložení nové smyčky do seznamu testuje, jestli se alespoň o jednu vazbu liší od předchozí.

Obsahová analýza

K analýze diagramů podle obsahu musíme na rozdíl od mechanického přístupu pochopit, jaký systém model reprezentuje a do procesu tím vstupuje jako hlavní prvek lidský faktor. Ten zároveň plní roli analytického stroje a přebírá roli počítače při vyhodnocení.

Práce na obsahové analýze mohou být založeny na snaze identifikovat mezi proměnnými ty podstatné, jako to dělá ve své práci [Beck et al, 2012], který v modelu oceňuje vlivy každé proměnné na ostatní a dále vliv času na proměnné, obě nalezené matice spojí a hledá, které proměnné jsou pro vývoj systému klíčové a které ne. Jeho cílem je poskytnout osobám, které realizují nad modelem rozhodnutí, efektivnější možnosti, jak systém ovlivňovat.

Dále se v mé práci tímto odvětvím nezabývám.

Praktická část

Výběr programovacího prostředí

Pro praktické zpracování algoritmů o kterých práce pojednává, jsem zvolil prostředí, ve kterém mám zkušenosti s objektovým programováním. Je jím programovací jazyk Visual FoxPro ve verzi 9.0, které disponuje souborem funkcí pro čtení a zápis souborů a také rozsáhlým souborem funkcí pro práci s dvourozměrnými poli.

Struktura formátu MDL

Seznámíme se dále s formátem souboru MDL. Jedná se o proprietální formát programu Vensim, který je dobře zdokumentovaný [Vensim 12]. Na každém řádku je zapsána

informace o jednom prvku diagramu. Na začátku každého řádku je první rozlišovací znak, který určuje typ a strukturu celého záznamu.

Soubor je tak velmi dobře strojově čitelný.

Každý řádek v diagramu přečteme a podle typu záznamu určíme jestli záznam budeme dále zpracovávat. Formát MDL umožňuje zapsat i další prvky, které nejsou v diagramech používány.

Zpracovávat budeme dříve diskutované části - uzly, vazby a smyčky.

Struktura diagramu vypadá takto:

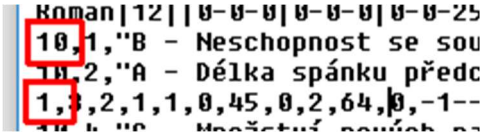
```
\\\---/// Sketch information - do not modify anything except names
V300  Do not put anything below this section - it will be ignored
*View 1
$192-192-192,0,Times New
Roman|12||0-0-0|0-0-0|0-0-255|-1--1--1|-1--1--1|96,96,170,0
10,1,"B - Neschopnost se soustředit na práci",414,48,61,25,3,131,0,0,0,0,0,0
10,2,"A - Délka spánku předchozí noci",128,104,54,25,3,131,0,0,0,0,0,0
1,3,2,1,1,0,45,0,2,64,0,-1--1--1,|12||0-0-0,1|(231,55)|
10,4,"C - Množství nových naléhavých úkolů",398,344,64,26,3,131,0,0,0,0,0,0
10,5,"E - Pocit hladu",649,80,40,20,3,3,0,0,0,0,0,0
1,6,5,1,1,0,43,0,2,64,0,-1--1--1,|12||0-0-0,1|(563,29)|
10,7,"F - Paralýza schopnosti jít se najíst",637,300,48,26,3,131,0,0,0,0,0,0
1,8,4,7,1,0,43,0,2,64,0,-1--1--1,|12||0-0-0,1|(509,346)|
1,9,7,5,1,0,43,0,2,64,0,-1--1--1,|12||0-0-0,1|(686,191)|
10,10,"D - Sklon k zanedbání prioritizace úkolů",411,177,61,24,3,131,0,0,0,0,0,0
1,11,10,4,1,0,43,0,2,64,0,-1--1--1,|12||0-0-0,1|(377,259)|
1,12,1,10,1,0,43,0,2,64,0,-1--1--1,|12||0-0-0,1|(400,126)|
1,13,4,2,1,0,43,0,2,64,0,-1--1--1,|12||0-0-0,1|(130,271)|
///---\\\
```

Obrázek 10 - část diagramu s informacemi o grafické reprezentaci diagramu

Vidíme, že zápis jednotlivých prvků v nákresu je čitelný. Každý prvek se vyskytuje na zvláštním řádku a jednotlivé parametry prvku jsou zapisovány oddělené čárkou. To je zásadní pro další strojové zpracování.

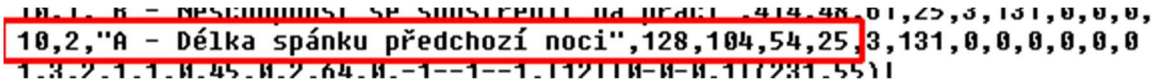
Identifikace prvků v diagramu

Prvky jsou v diagramu rozlišené hodnotou od začátku řádky k oddělovači (čárka).



Obrázek 11 - rozlišení typu prvku

Uzly jsou určeny tím, že řádek začíná znaky “10,” a zápis má strukturu (odděleno čárkou)

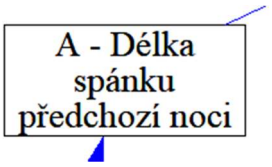


Obrázek 12 - zápis uzlu v souboru MDL

10 - typ záznamu (uzel)

2 - interní identifikátor

“A - Délka….” - textový název uzlu



128,104 - pozice X,Y souřadnice pravého spodního okraje obdélníku s textem proměnné

54,25 - šířka a výška obdélníku s textem proměnné

Vazby jsou určeny tím, že řádek začíná znaky “1,” a zápis má strukturu (odděleno čárkou)

```
1,6,5,1,1,0,43,0,2,64,0,-1--1--1,|12||0-0-0,1|(563,29)|
```

Obrázek 13 - zápis vazby v souboru MDL

1 - typ záznamu (vazba)

6 - interní identifikátor

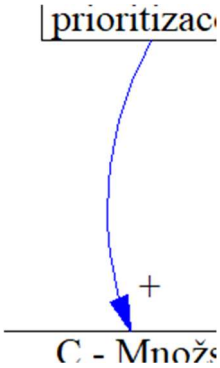
5 - interní identifikátor uzlu ze kterého vazba vychází

1 - interní identifikátor uzlu do kterého vazba směřuje

1 - tvar čáry (kruh, polygon, přímka)

0 - příznak zda skrýt vazbu

43 - ASCI znak polarity (povolené znaky s/S/+/o/O/-



… další atributy pro naši práci nepodstatné

Smyčky jsou určeny tím, že řádek začíná znaky “12,” a zápis má strukturu (odděleno čárkou)

```
12,15,0,223,73,40,40,4,135,0,3,0,0,0,0,255-0-0,0-0-0,|0||255-0-0
Missing loop - type is B
```

Obrázek 14 - zápis smyčky v souboru MDL

12 - typ záznamu komentář (zde použito pro smyčky)

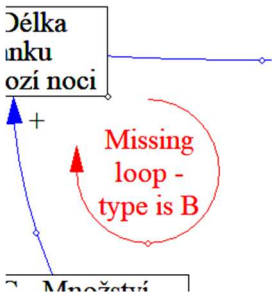
15 - interní identifikátor

0 - index ikony u komentáře

223,73; - pozice X,Y souřadnice středu komentáře

40,40 - šířka a výška

4 - tvar čáry (zde kruh)



… další atributy (fonty, barvy, zarovnání)

“Missing loop - type is B” - text u komentáře, je na novém řádku

Algoritmus validace diagramu

Uzly

Při konstrukci algoritmu postupujeme od nižších částí diagramu, kterými jsou uzly. Zjistíme jejich počet a uložíme si jejich seznam. U každého uzlu zjišťujeme interní číslo v původním diagramu (jedinečné), přiřadíme jedinečné interní číslo pro naše zpracování a uložíme si název a souřadnice kde každý uzel v diagramu leží.

U uzlů žádné další kontroly neprovádíme.

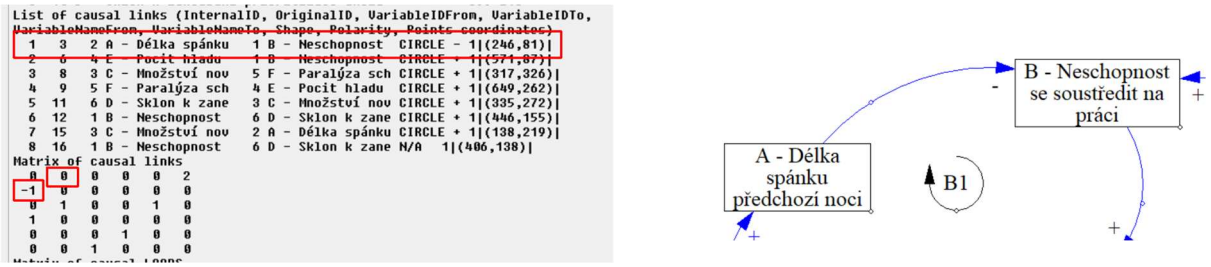
Vazby

Dále se zabýváme zkoumáním vazeb. Stejně jako u uzlů zjistíme počet vazeb a odkud a kam vazby vedou. V seznamu, který si z vazeb tvoříme, ukládáme původní identifikátor vazby a dále příznaky odkud a kam vazba vede. Zde již ukládáme pouze naše interní identifikátory uzlů. Také zjistíme, zda jsou vazby posilující (označení vazby je “S / s / +”) nebo oslabující “O / o / -” a pokud označení chybí nebo je mimo povolené hodnoty, vazbu vyhodnotíme jako chybnou.

Ze seznamu vazeb zpracujeme matici sousednosti, kde jsou v řádcích jednotlivé uzly, počet uzlů odpovídá počtu řádků i sloupců, matice je čtvercová. Směr vazby je vyznačen tak, že řádky matice jsou zdrojové body “Z jakého uzlu vazba vychází” a sloupce jsou cílové body “DO jakého uzlu vazba směřuje”.

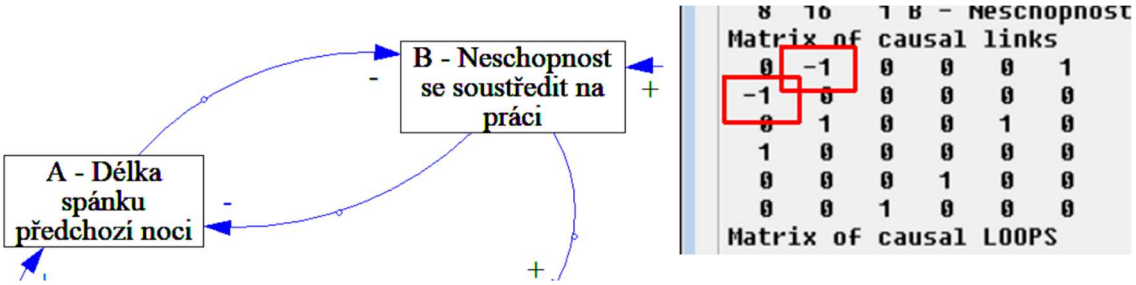
Matice sousednosti

Vazba na příkladu vede z uzlu s interním číslem 2 (A - Délka spánku...) do uzlu s interním číslem 1 (B - Neschopnost se soustředit…) a v matici sousednosti je tedy na řádku 2 (odkud vazba vede) ve sloupci 1 (kam vazba směřuje) hodnota -1, která značí oslabující vazbu.



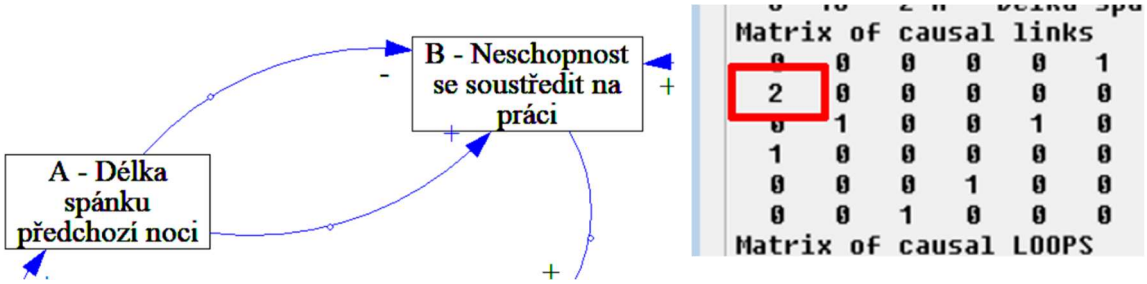
Obrázek 12 - vazba a její pozice v matici

Dále je také vidět, že v matici v řádku 1 a sloupci 2 je 0. Pokud by vedla vazba také zpět z bodu 1 do bodu 2, což je povoleno, tak by i v tomto bodě [1,2] bylo také nenulové číslo, vyjadřující vazbu opačným směrem. Jak vypadá reprezentace v matici je vidět na obrázku 13.



Obrázek 13 - 2 vazby mezi 2 uzly

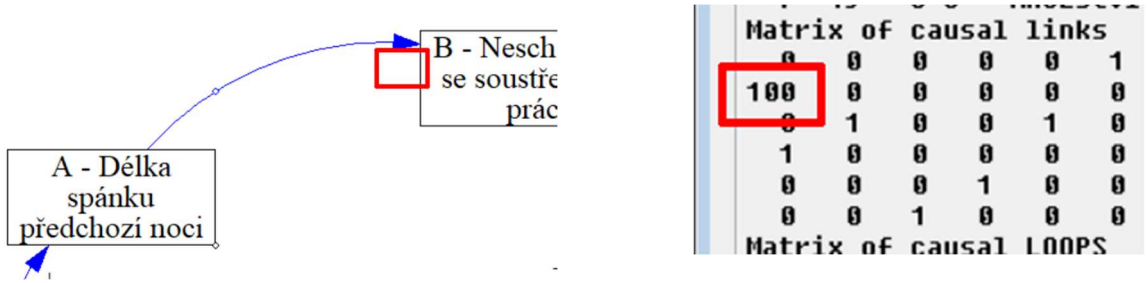
Je třeba uvést ještě jeden speciální scénář, kdy z jednoho uzlu do druhého vede více než jedna vazba. Tento stav diagramu program Vensim s upozorněním umožňuje, ale v příčinných smyčkových diagramech není takový zápis povolen. Náš algoritmus vyhodnocuje počet vazeb větší než 1 mezi dvěma uzly jako chybu a do matice zaznamená absolutní hodnoty každé vazby tak, aby nemohlo dojít k sečtení hodnot 1 a -1, pokud by každá vazba měla jinou polaritu, což by vedlo k výslednému součtu 0 a



cílový stav by nebylo možné odlišit od stavu, kdy žádná vazba mezi uzly není. Situace je znázorněna na obrázku 14.

Obrázek 14 - dvojnásobná vazba mezi 2 uzly

Pro lepší vizuální odlišení chybné nebo chybějící polarity vazby používáme hodnotu 100 místo +/- 1.



Obrázek 15 - chybějící označení polarity vazby

Vzhledem k tomu, že další část algoritmu (identifikace smyček) přímo staví na zjištěné struktuře vazeb, je vhodné vypořádat chyby ve vazbách než se pustíme do kroků na vyšší úrovni.

Nalézání smyček

Pokračujeme v identifikaci smyček v diagramu. Smyčky jsou v diagramu zakreslené symbolem kruhové šipky a označením R nebo B a pořadovým číslem. Smyčky nejsou nijak provázané s vazbami v diagramu jako je tomu u uzlů a vazeb.

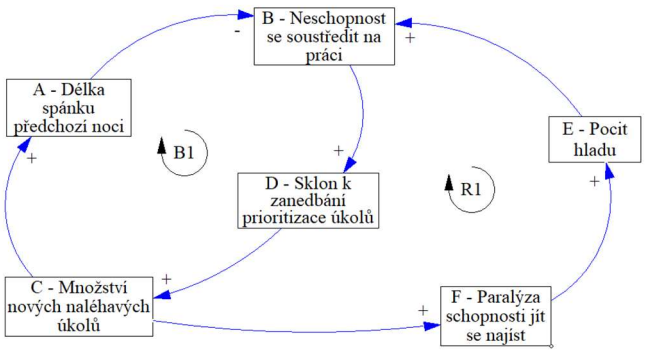
Pro zakreslení smyčky se používá prvek “komentář” a ten nemá vazbu na jiné objekty v diagramu. Zakreslení smyčky pouze graficky odpovídá tak, aby jej vazby, tvořící smyčku obklopovaly. Vzhledem k tomu, že při kontrole chceme prověřit, že zakreslené smyčky odpovídají vazbám, které je obklopují, ale také chceme zjistit, jestli některá smyčka nechybí, budeme postupovat tak, že nalezneme nejprve všechny smyčky, které v diagramu mohou existovat bez ohledu na to, jestli jsou v diagramu zakreslené nebo

ne a následně budeme zkoumat podle polohy vazeb a zakreslených smyček, jestli nalezené smyčky odpovídají zakresleným.

Tímto postupem můžeme očekávat dva výsledky. Prvním je "vše v pořádku" - pokud v diagramu nebude chyba. Druhým je, že rozdíl najdeme a v diagramu je nalezena chyba - buď smyčka chybí úplně nebo je nepřesně zakreslena - pozicí nebo označením polarity.

Smyčku definujeme tak, že při procházení mezi uzly ve směru působící vazby se dostaneme do bodu, kterým jsme již prošli. Seznam prošlých bodů se tak uzavře a je nalezena jedna ze smyček. Dalším krokem je vytvořit algoritmus, který nalezne všechny smyčky, které v diagramu existují.

Při hledání algoritmu musíme tedy nejprve zvolit vhodný způsob procházení. Zkusme položit tvrzení, že pokud prozkoumáme všechny vazby v diagramu, tak nemůžeme vynechat žádnou z existujících smyček. Můžeme tedy postupovat tak, že každou vazbu budeme považovat za zárodek smyčky a při procházení vazeb budeme postupně tvořit skupiny vazeb, které na sebe v uzlech navazují. Pokud takto projdeme všechny vazby, získáme některé smyčky. Na obrázku 16 si ukážeme několik kroků.



	A	B	C	D	E	F
A		-1				
B				1		
C	1					1
D			1			
E		1				
F					1	

Obrázek 16 - matice sousednosti diagramu

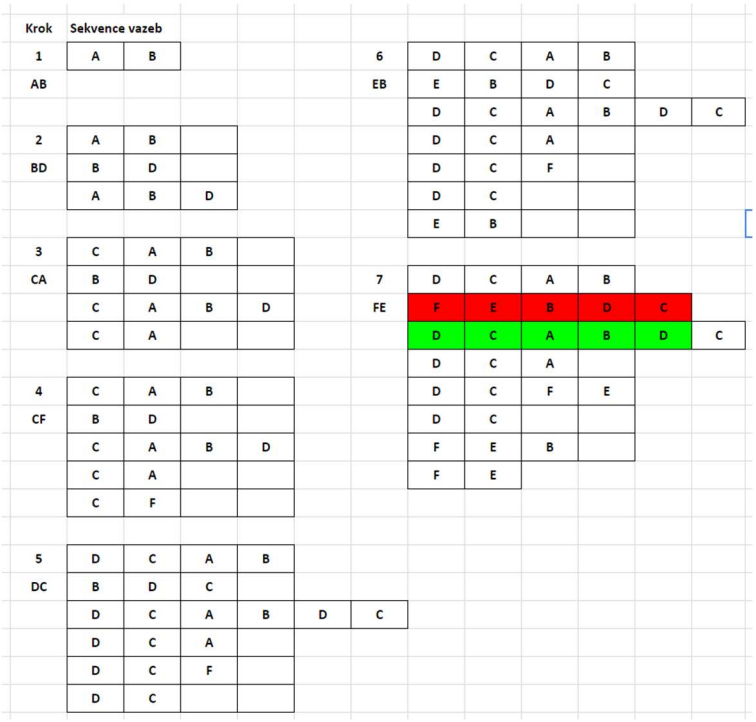
Průchod maticí začínáme po řádcích, ty nám znázorňují směr vazeb. Každou vazbu, kterou zpracujeme, položíme na nový řádek našeho seznamu "budoucích smyček".

Krok	Sekvence vazeb			
1	A	B		
2	A	B		
	B	D		
	A	B	D	
3	C	A	B	
	B	D		
	C	A	B	D
	C	A		

První vazba vede z A do B, tedy do výsledného pole píšeme A-B jako zárodek první smyčky. Pokračujeme v průchodu druhým řádkem, kde máme další zárodek smyčky B-D. V tuto chvíli ještě nemůžeme jít na třetí krok, protože jsme zatím zjistili, že máme dva zárodky smyček A-B a B-D, ale my především vidíme, že v tuto

chvíli můžeme rozšířit vazbu A-B o B-D na jejím konci zde dochází ke shodě v uzlech a po spojení se z A-B a B-D stane A-B D. Na konci druhého kroku máme tedy 3 zárodky smyček A-B, B-D a A-B-D.

Postupujeme do dalšího kroku, kde máme na 3. řádku C-A. Opět se snažíme, abychom ve všech řádcích našeho seznamu, kde "pěstujeme zárodky smyček" měli o C-A rozšířené všechny naše smyčky kde to lze. Řada A-B-D i A-B lze rozšířit tak, že C-A předřadíme a vznikne nám C-A-B-D resp. C-A-B, řadu B-D nemůžeme rozšířit, a jako poslední zárodek položíme samotnou vazbu C-A. Stav po 3. kroku vidíme na obrázku:



Obrázek 17 - stav smyček po 7. kroku

Dosavadní postup můžeme tedy shrnout do těchto kroků:

1. Vezmi novou vazbu a udělej z ní samostatný řádek
2. Podívej se, jestli vazbu lze spojit přes shodné jméno uzlu s koncem nebo začátkem jiného řádku
3. Prověř takto pro všechny dosud nalezené řádky
4. Celé opakuj pro všechny nenulové prvky v matici

Matici projdeme až do konce a zjistím, že jsme nalezli jednu smyčku (na obrázku 17 zeleně), ačkoli v diagramu jsou smyčky 2.

Při pohledu na krok 7 vidíme, že druhou smyčku (červeně) máme téměř uzavřenou, zbývá pouze vazba CF. Tím, jak matici procházíme, jsme vždy schopní bezpečně

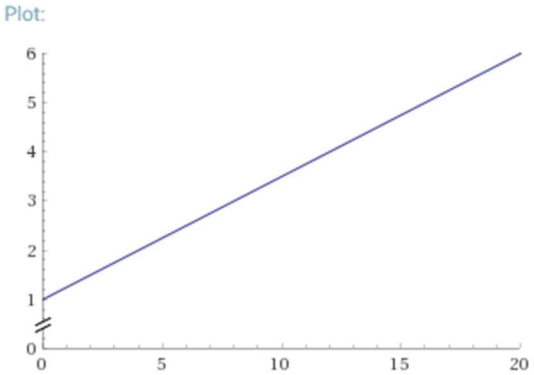
dosáhnout spojení všech vazeb, ze kterých se každá smyčka skládá, ale při zvyšování počtu uzlů ve smyčce vzroste pravděpodobnost, že zpracovávaná vazba (v našem případě C-F) v okamžiku, kdy probíhá zpracování (zde krok č.4) ještě nemá zárodek který by končil na C nebo začínal na F. Tyto vazby se vkládaly až v 5. resp 7. kroku.

Pokud bychom celý cyklus zopakovali, vazbu bychom na své místo doplnili v dalším průchodu. To je ale poměrně náročné na výkon a pokud bude vzrůstat počet uzlů ve vazbě, nemusí ani jeden další průchod stačit. Pokud vezmeme příklad kde je ve smyčce např. 12 uzlů, které jsou maximálně nevýhodně uspořádány, tak se mi může stát, že se mi v prvním průchodu podaří ze smyčky postavit pouze 3 sousední vazby (bude to případ, kdy jsou procházení vazby vždy "objednu" a vznikají tak osamocené zárodky smyček, na které se ve 4 případech podaří z každé strany zárodku připojit navazující vazby) a v dalším cyklu pak další 2 vazby (protože se číslo udávající pořadí střídá "menší, větší" z každé strany, to je celkem 7 vazeb a budeme potřebovat třetí průchod na poslední vazbu.

Počet průchodů diagramem

Konstrukce funkce je tedy x - počet uzlů ve smyčce, první průchod zpracuje 3 vazby, poslední 1 vazbu, a pak při každém průchodu 4 vazby. Můžeme tedy zapsat jako:

$$f(x) = 2 + \frac{x-4}{4}$$



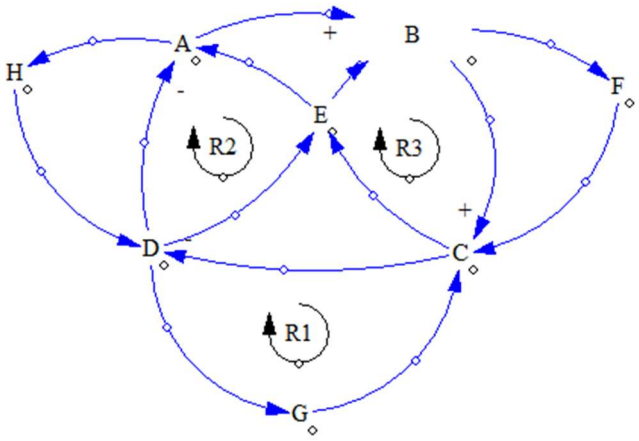
Protože dopředu nevíme, kolik maximálně bude ve smyčce uzlů, můžeme provést první zpracování a při druhém sledovat, jestli jsme nějaké vazby při druhém průchodu doplnili.

Pokud ano, zpracování znovu opakujeme, až nebude při průchodu doplněna žádná smyčka o chybějící vazby, průchod ukončíme. Nabízí se i další metody řešení, kde budeme vícenásobně zpracovávat pouze vazby, které mají potomky, u kterých tato situace může nastat. Tato optimalizace bude rozpracována v budoucí verzi programu.

Pro náš diagram, ve kterém máme 6 proměnných, budeme tedy potřebovat 2+(6-4)/4=2,5 tedy 3 průchody.

Speciální typy smyček

Dále si povšimneme speciálních scénářů, které mohou dále v diagramu nastat (obrázek 18):

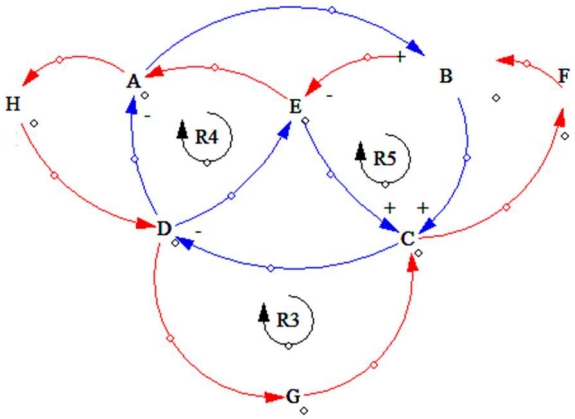


Obrázek 18 - speciální případy smyček

1. Dvě smyčky spolu sousedí jednou vazbou (R1)
2. Dvě smyčky se překrývají tak, že překryv netvoří třetí smyčku (R2)
3. Dvě smyčky se překrývají tak, že překryv tvoří další smyčku (zahrnuje také všechny případy kdy jedna smyčka je podmnožina jiné, R3)

Program, který v rámci této práce vzniká, je v této fázi je již spolehlivě rychlejším pomocníkem pro ověření nejsložitějších smyček a scénářů než ruční identifikace.

Například máme diagram na obrázku 19, kde je 8 proměnných a program pro počet průchodů rovno 2 nenalezl nejdelší smyčku (vyznačena červeně), ale pro počet průchodů 3 již nalezl i tuto nejdelší smyčku.



Obrázek 19 - Smyčka v diagramu s 8 uzly a 3x průchod diagramem

Tímto uzavřeme část algoritmu o vyhledání smyček a v další části se budeme zabývat určením typu smyčky.

Typ smyčky

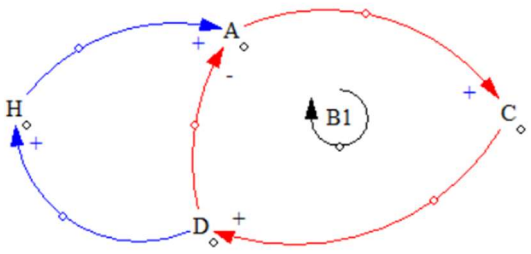
Typ smyčky určujeme podle počtu oslabujících vazeb ve smyčce. Posilující smyčka obsahuje sudý počet oslabujících vazeb, značíme ji "R" a pořadovým číslem tohoto typu smyčky v diagramu. Rovnovážná smyčka obsahuje lichý počet oslabujících vazeb a označujeme ji "B" a pořadovým číslem tohoto typu smyčky v diagramu. Nalezené smyčky označíme dle těchto pravidel a získáme konečný seznam smyček identifikovaných v diagramu, což je pro další zpracování první ze dvou stran, které budeme porovnávat.

Smyčky zakreslené a vypočtené

Porovnávací stranu získáme ze seznamu smyček zakreslených v diagramu. Oba seznamy srovnáme a naším cílem je zjistit, jestli každá ze zakreslených vazeb v diagramu odpovídá jedné ze zjištěných vazeb programem.

Je zřejmé, že nemůžeme smyčky porovnávat podle jména (označení R1, B1…), protože nevíme, jestli tvůrce diagramu postupoval stejným způsobem při identifikaci smyček jako my v programu a koeficienty 1…X přiřadil smyčkám shodně.

Využijeme toho, že v diagramu je smyčka zakreslena vždy uprostřed vazeb, které ji tvoří a je jimi obklopena. Příklad je na obrázku 20 červeně, kde vidíme smyčku B1, kterou tvoří body A, C a D.



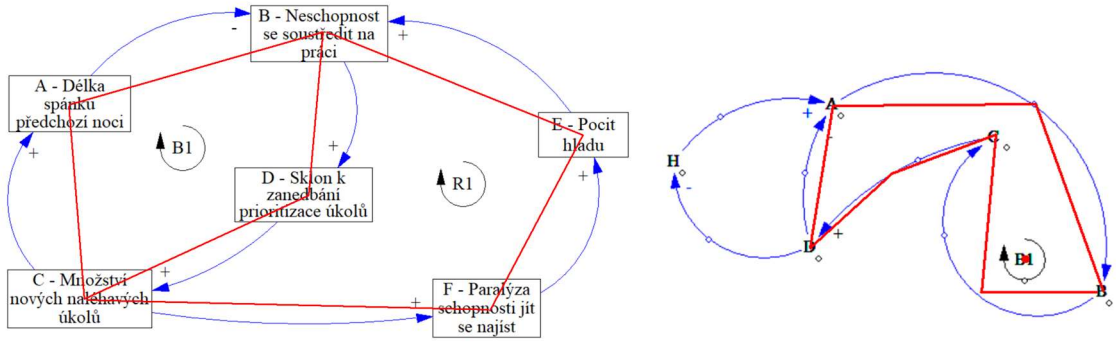
Obrázek 20 - pozice smyčky a vazeb

V tomto scénáři vezmeme v potaz souřadnice bodů A, C a D a získáme trojúhelník, které tyto 3 body vyznačují. V komplikovanějším diagramu se na smyčce podílí více vazeb než 3, což znamená řešit úlohu obecně pro polygon.

Komplikaci přináší také fakt, že vazby nejsou znázorněny přímkami, ale částí kruhu. Vzhledem k tomu, že v diagramu se smyčky zakreslují do nejvíce do středu polygonu vymezeného jednotlivými uzly, není příliš pravděpodobné, že by zakreslené smyčky

bylo v okrajové části některého z kruhů, které smyčku obepínají. Rozhodl jsem se aproximovat při výpočtu kruhové výseče tvořící smyčky úsečkami mezi dvěma uzly.

Nalezená smyčka je programem vyhodnocena jako správná pokud se vyskytuje uvnitř polygonu tak jak je znázorněno na obrázku 21 vlevo, obě smyčky podmínku splňují. Vpravo je zakreslen komplikovanější tvar smyčky, kde je nezbytné použít polygon s počtem hran vyšším než počet uzlů, v praxi se tento způsob zákresu však příliš nevyskytuje. V programu tak není realizován.



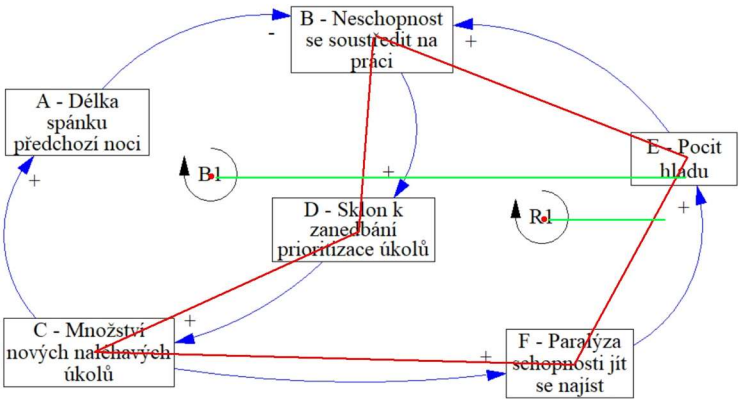
Obrázek 21 - ohraničení smyčky mnohoúhelníkem mezi uzly bez lomené čáry a s lomenou čárou

Pro určení, zda je bod uvnitř polygonu, existuje několik algoritmů.

Pro implementaci jsem zvolil metodu Jordan Curve Theorem, která říká: (Haines 2001) Pokud paprsek promítnutý ze zkoumaného bodu vodorovně protne zkoumaný polygon v lichém počtu míst, leží zkoumaný bod uvnitř polygonu. Sudý počet protnutí znamená, že bod leží vně polygonu.

Postupujeme tedy tak, že získáme rovnice všech hran polygonu. Dále použijeme souřadnice zakresleného bodu se symbolem smyčky. Budeme testovat, jestli taková úsečka tvořena vrcholy sousedních uzlů protíná Y souřadnici zkoumaného bodu (Finley 2007). Pokud ano, zajímé nás, jestli takový bod protnutí leží vlevo nebo vpravo od zkoumaného bodu. Provedeme pro všechny úsečky a pokud máme na jedné straně (vpravo nebo vlevo) sudý počet protnutí, tak je zkoumaný bod mimo polygon, v ostatních případech je uvnitř polygonu.

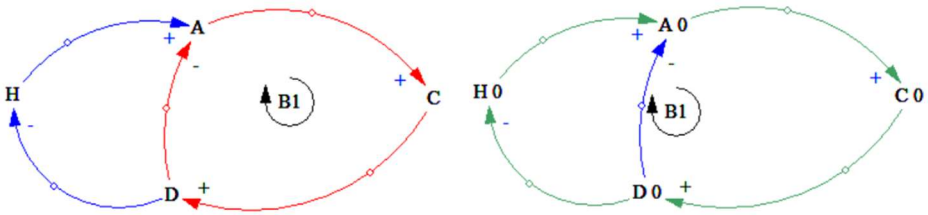
Na obrázku 22 je možné vidět, že zelený paprsek z smyčky B1 protne polygon 2x a je tak vně polygonu a paprsek R1 protne polygon 1x (jedním směrem) a leží tedy uvnitř polygonu.



Obrázek 22- určení zda je bod uvnitř polygonu nebo vně

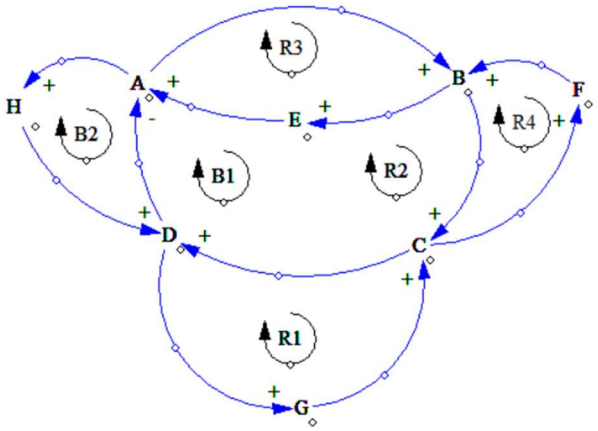
Máme tedy pro každou nalezenou smyčku v diagramu určeno, ve kterých zjištěných smyčkách, reprezentovaných odpovídajícími polygony, se nachází.

Nyní rozebereme komplikovanější scénáře, které mohou nastat. Na obrázku 23 máme v diagramu nalezenou ještě jednu smyčku A0, C0, D0 H0, která rovněž obklopuje smyčku B1 (na obrázku 23 zeleně). Vzhledem k tomu, že se smyčky mohou překrývat, není na obrázku 23 jednoznačně možné určit, kterou smyčku autor diagramu myslel, když do diagramu zakreslil pouze jednu smyčku, ačkoli jsou v diagramu smyčky dvě. Je zjevné, že tento příklad se může dále komplikovat a může se takto překrývat 3 a více smyček a bude třeba celý problém řešit obecně.



Obrázek 23 - nekolik zjištěných smyček k jedné nalezené

Budeme při určování dvojic smyček postupovat tak, že pro každou zakreslenou (označujeme také “nalezenou”) smyčku zjistíme, jestli je svými souřadnicemi součástí některé ze zjištěných smyček. Pracovat budeme s modelem na obrázku 24:



Obrázek 24 - diagram s překrývajícími se smyčkami

Párování smyček

Postupně zapisujeme do matice, která má ve sloupcích zjištěné smyčky a v řádcích nalezené smyčky. Nalezená smyčka R1 je například součástí zjištěné smyčky CDG a smyčky AHDGCFBE, v matici znázorníme na obrázku 25 červeně. V dalším kroku identifikujeme R2 a B1 (podobné členství, tedy zde jeden krok) jako členy smyčky ABCD a taktéž velké AHDGCFBE, na obrázku zeleně. Smyčka R3 je obsažena ve zjištěné smyčce ABE a také ABCD, modře, a R4 a B2 jsou členy AHDGCFBE a AHD (B2) a BCF (R4).

	Smyčky zjištěné programem VerifyCLD					
Nalezené smyčky v diagramu	ABCD	ADH	ABE	BCF	CDG	AHDGCFBE
R1					1	1
R2	1					1
B1	1					1
R3	1		1			
R4				1		1
B2		1				1

Obrázek 25 - matice možných umístění zjištěných a nalezených smyček

Dále postupujeme tak, že procházíme všechny nalezené smyčky a testujeme, zda je nalezená smyčka pro některou zjištěnou smyčku jedinou, která je v ní obsažena. Taková dvojice je pak jednoznačná, zde např. R1 pro CDG, R3 pro ABE, R4 pro BCF

nebo B2 pro ADH a a můžeme ze souboru odstranit nalezené i zjištěné smyčky, viz obrázek 26.

	Smyčky zjištěné programem VerifyCLD					
Nalezené smyčky v diagramu	ABCDB	ADHB	ABER	BCFR	CDGR	AHDGCFBER
R1					1	1
R2	1					1
B1	1					1
R3	1		1			
R4				1		1
B2		1				1

Obrázek 26 - matice zbylých možných přiřazení, po identifikaci jednoznačných párů

Po dokončení zpracování mohou nastat tyto scénáře:

1. Zbývají některé zjištěné smyčky ale žádná nalezená - v diagramu nejsou některé smyčky vůbec zakresleny
2. Zbývají některé nalezené smyčky, ale žádná zjištěná - v diagramu jsou zakreslené smyčky, které neexistují
3. Zbývají nalezené smyčky (jedna nebo více takových skupin), které jsou součástí více zjištěných smyček - musíme dále pokračovat v přiřazení podle typu smyčky (R nebo B), pozor, bod 3 zpracujeme odděleně od bodu 4, protože zde se nejedná o chybu v diagramu ale pouze o nejednoznačnost, v bodě 4 se naopak o chyby jedná
4. Zbývají nalezené smyčky a zjištěné smyčky a nalezené smyčky nejsou součástí žádných zjištěných smyček - jedná se o nepřesnost zakreslení nalezených smyček a v identifikaci párů pokračujeme podle typu smyčky (R nebo B)

V našem případě nám zbývá jedna skupina kde jsou 2 nalezené a 2 zjištěné smyčky, tedy budeme pracovat se scénářem 3. Budeme se snažit přiřadit smyčky podle typu. Vidíme, že nalezené smyčky jsou jedna R a jedna B. Zjištěné smyčky jsou také jedna R a jedna B, tedy přiřazení je jednoznačné a výsledný soubor je na obrázku 27.

Nalezené smyčky v diagramu	Smyčky zjištěné programem VerifyCLD					
	ABCDB	ADHB	ABER	BCFR	CDGR	AHDGCFBER
R1					1	1
R2	1					1
B1	1					1
R3	1		1			
R4				1		1
B2		1				1

Obrázek 27 - výsledný soubor po identifikaci všech párů smyček

Označení chybných smyček

Body 1 a 2 zpracujeme tak, že smyčky, které zbyly, jsou označeny jako chybné.

Bod 3 zpracujeme tak, že vždy pro každou skupinu najdeme nalezenou smyčku stejného typu mezi jakýmikoli zbývajícími zjištěnými a pokud se nám to povede, soubor o tyto páry zjednodušíme a postup opakujeme.

Na konci zpracování nám může zbýt

A. Nenulový počet nalezených a nenulový počet zjištěných smyček - jedná se o chybný typ zakreslené smyčky (R versus B), tyto páry označíme chybou a odebereme ze souboru, opakujeme než počet zbývajících nalezených nebo zjištěných (stále dodržujeme případné skupiny ve kterých tyto překryvy vznikly, může jich v souboru být i více) smyček klesne na nulu, poté pokračujeme bodem 2

B. Počet nalezených nebo zjištěných smyček je nula - tyto přebytečné smyčky ze všech případných skupin zahrneme do zpracování nalezených smyček, které nejsou součástí zjištěných smyček (bod 4 v předchozím zpracování)

Bod 4 zpracujeme tak, že se mezi všemi zbývajícími nalezenými smyčkami snažíme podle typu vazby nalézt odpovídající smyčku mezi zjištěnými smyčkami a opět soubor při každém nálezu redukujeme, dokud neklesne počet nalezených nebo zjištěných smyček na nulu. Zbývající smyčky zpracujeme dle bodu 1 nebo 2.

Tímto krokem je identifikace nalezených smyček, které autor diagramu zakreslil, dokončena. Ke každé zakreslené smyčce existuje jednoznačně odpovídající vypočtená vazba nebo taková vazba v diagramu není nebo je v diagramu umístěna jinde. Jsou tak pokryty všechny scénáře, které mohou nastat.

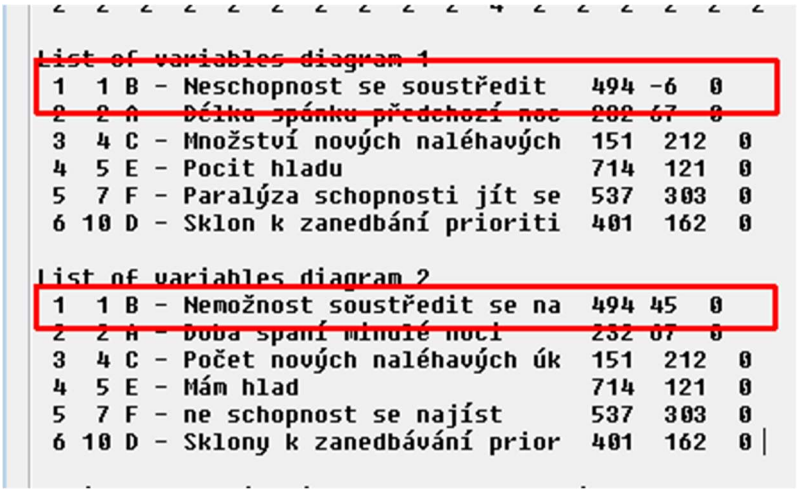
Algoritmus porovnání 2 diagramů

Při porovnání dvou diagramů začneme na nejnižší úrovni, tedy u uzlů. Předpokládáme, že pokud jsou tvořeny 2 diagramy, které řeší jednu problematiku, budou se i jejich uzly podobně jmenovat. Některé mohou chybět, jiné přebývaat, ale měli bychom být schopni stanovit určitý průnik. Postupujeme tak, že ze všech názvů uzlů vybereme slova delší než 2 znaky tak, aby porovnání nebylo zkresleno různými spojkami, zájmeny apod. Sestavíme seznam těchto slov a vytvoříme matici, kde v řádcích jsou slova z prvního diagramu a ve sloupcích slova ze druhého diagramu. Matici procházíme pro všechny prvky a hodnotíme podobnost slov takto:

1. 2 zcela shodná slovo - 4 body
2. Jedno slovo je obsaženo ve druhém, např. schopnost X schopnosti - 2 body
3. Obě slova mají společnou část delší než 5 znaků, ale od určitého znaku se odlišují, např. spánku X spánek - 1 bod
4. Slova zcela různá nebo shodující se do 5 znaků délky - mínus dva body

Koeficienty byly vybrány experimentálně na základě srovnání 5 dvojic diagramů tak, aby poskytovaly pro tento vzorek nejlepší výsledky. Bodem 4 je dán důraz na to, aby raději algoritmus shodu nenašel, než aby se zmýlil a vykázal falešný pozitivní výsledek.

Další zpracování probíhá tak, že pro všechny proměnné projdeme slova, ze kterých se skládají. Pokud je dvojice slov ohodnocena kladnými body, přičtu co celkového hodnocení páru proměnných. Pokud není nalezena dvojice s kladným ohodnocením, vyberu nejmenší záporné ohodnocení, takto opakuji pro každou proměnnou právě jednou. Za každé slovo tedy mohu získat právě jednou kladné nebo záporné ohodnocení. Na příkladu vidíme dvě proměnné.



Při výpočtu získáme slova:

```
List of words in diagram 1
 1  3 Neschopnost
 1  5 soustředit
 1  7 práci
```

```
List of words in diagram 2
 1  3 Nemožnost
 1  4 soustředit
 1  7 práci
```

Ohodnotíme v matici podobnost jednotlivých slov:

```
Matrix of words diagram 1 (rows) / diagram 2 (columns)
-2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2
-2  4 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2
-2 -2  4 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2
-2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2
-2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2
```

Výpočtem výše dojdeme k výsledku (jsou brány jen nejpodobnější 3 páry pro jednoduchost)

1. neschopnost X nemožnost - mínus 2 body, index [1,1]
2. soustředit X soustředit - plus 4 body, přesná shoda, index [2,2]
3. práci X práci - plus 4 body, přesná shoda, index [3,3]

Součet je plus 6 bodů, také vidíme ve výsledné matici:

```
Variables matrix diagram 1 (rows) / diagram 2 (columns)
 6 -8 -8 -6 -6 -6
-8 -2 -8 -8 -8 -8
-8 -8 10 -8 -8 -8
-6 -8 -8  0 -4 -4
-8 -8 -8 -8  2 -8
-6 -8 -8 -6 -6  1
```

Takto postupujeme pro všechny proměnné v obou diagramech, počet průchpdů je MxN.

Výslednou matici procházíme tak, že najdeme největší prvek v matici, pro nás zde hodnota 10. Případy, kdy by vycházelo více kladných hodnot stejných budeme realizovat v budoucí verzi programu. Souřadnice tvoří přímo interní identifikátory obou proměnných a máme tak vytvořený první pár. V dalším kroku označíme tuto proměnnou (ve sloupci i řádku) za zpracovanou, sloupec i řádek matice nulujeme. Takto pokračujeme dále pro další proměnné dokud máme nějaké prvky matice větší než nula.

Výsledkem jsou vytvořené páry B-B, C-C, F-F a D-D:

```
Matrix of final variable pairs from 2 diagrams
 1  0  0  0  0  0
 0  0  0  0  0  0
 0  0  1  0  0  0
 0  0  0  0  0  0
 0  0  0  0  1  0
 0  0  0  0  0  1

Pairs of variables from 2 diagrams

B - Neschopnost se soustředit na práci -- B - Nemožnost soustředit se na práci

C - Množství nových naléhavých úkolů -- C - Počet nových naléhavých úkolů

F - Paralýza schopnosti jít se najíst -- F - ne schopnost se najíst

D - Sklon k zanedbání prioritizace úkolů -- D - Sklony k zanedbávání prioritizace
```

Dále pokračuji ve srovnání vazeb mezi identifikovanými proměnnými a pokud se shodují, jsou rovněž označeny a spárovány. Smyčky je možné spárovat pouze pokud jsou identifikovány a spárovány všechny vazby, které smyčku tvoří. Tím je kontrola 2 diagramů dokončena.

Použité funkce

Při programování kódu jsem použil tyto hlavní komponenty a funkce pro práci s obsahem souboru MDL:

```
STRCONV(FILETOSTR(lcFileName),11)
```

Celý obsah souboru program načte do paměti, kde je možné obsah dále zpracovávat a procházet.

```
OCCURS(CHR(13) + CHR(10) + "10,"
```

Textové pole je prohledáno na výskyt konkrétního souboru znaků, kterým je identifikován nový řádek v souboru. Od takové pozice čteme dál do konce řádku a taková sekvence znaků je pro nás jeden prvek v diagramu.

```
FOR lncount = 1 TO lnCountRows
     ** check if this is variable
     IF LEFT(lcTmpRows[lnCount,1],3) = "10,"
          lcTmpVars[lnCountVar,1] = lnCountVar
```

Cyklus, kterým se do dvourozměrného pole vkládají jednotlivé objekty diagramu, pro každý prvek (řádek) rozlišujeme různé atributy prvku (identifikátor, název, souřadnice) a tyto zapisujeme do jednotlivých sloupců pole.

```
LEFT(lcTmpRows[lnCount,1],3) = "10,"
```

Pokud řádek začíná sekvencí “10,” je to v diagramu “uzel”. Pro řádek začínající “1,” se jedná o vazbu:

```
LEFT(lcTmpRows[lnCount,1],3) = "1,"
```

Procházíme všechny vazby a pro každou zapisujeme do sousedné matice polarity jednotlivých vazeb.

```
DO CASE
      CASE INLIST(lcTmpCausL[lnCount,8],"+","s","S")
           lnMatrixVal = 1
      CASE INLIST(lcTmpCausL[lnCount,8],"-","o","O")
           lnMatrixVal = -1
```

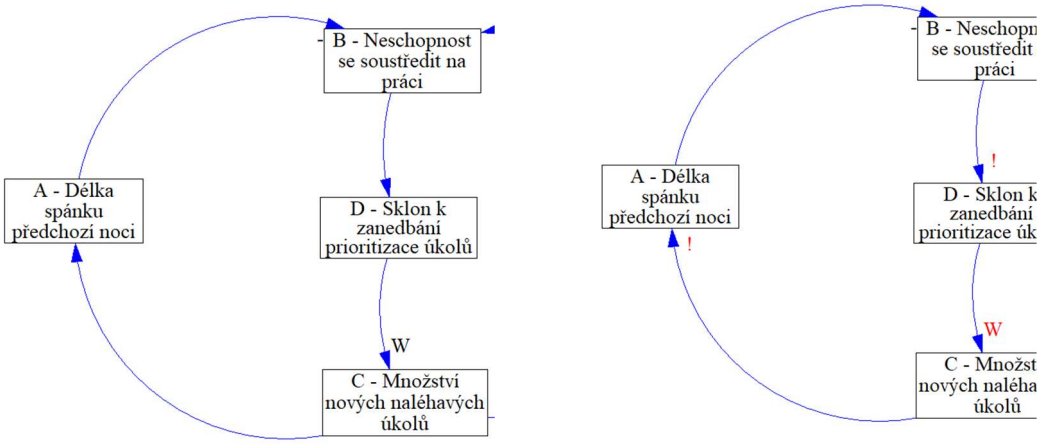
Další kontrukty programoacího jazyka se již opakují a jejich prezentace zde není nutná.

Prezentace výstupu

Při zpracování program vytvoří kopii diagramu a v ní zvýrazní nalezené chyby.

Chybějící nebo nesprávné označení vazby

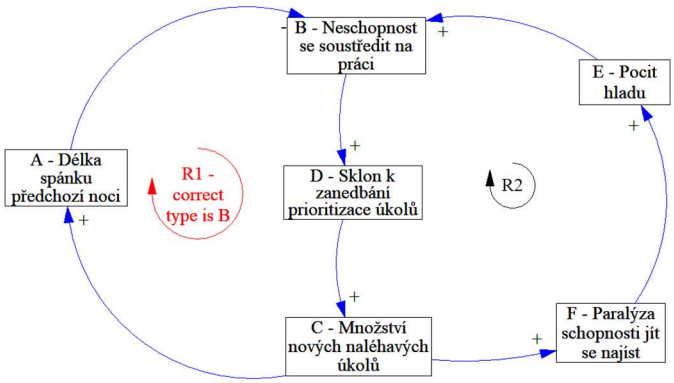
Pokud vazba není označena jedním z povolných znaků s/S/+/o/O/-, nebo není označena vůbec, program chybějící polarity označí červeným vykřičníkem a nesprávná označení označí červeně:



Obrázek 28 - vlevo chybějící polarity a vpravo korekce z programu

Nesprávně zvolený typ smyčky

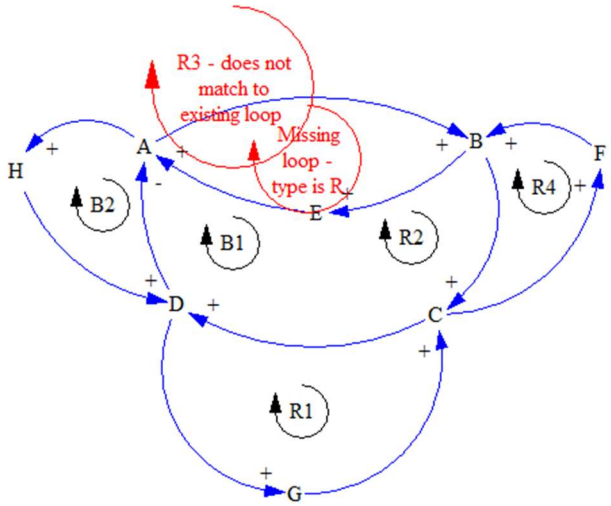
Pokud program vyhodnotí na základě nalezených smyček a polarity vazeb, které je tvoří, že je některá smyčka nesprávně určená (R místo B nebo naopak), je chybně identifikovaná smyčka označena červeně a naznačen správný typ smyčky.



Obrázek 29 - smyčka na správném místě ale nesprávně nazvaná - R místo B

Smyčka zakreslena na chybném místě

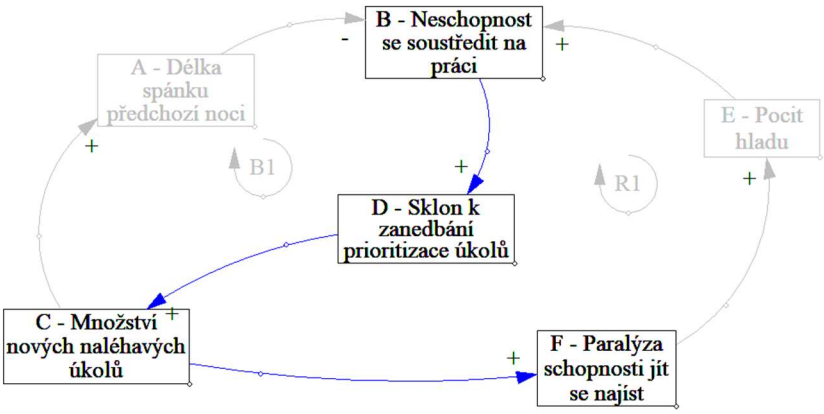
Pokud je smyčka zakreslena tak, že svým středem není uvnitř polygonu tvořeného jednotlivými uzly identifikované smyčky, je vyhodnocena jako chybějící a znázorněna na správném místě, při výpočtu je použita metoda výpočtu centroidu polynomu. Původní smyčka je tím pádem označena jako nespárovaná a je rovněž označena červeně jako chybná. Pro opravu stačí smyčku posunout na správné místo.



Obrázek 30 - smyčka na místě, kde nemá být (vlevo) a chybějící smyčka (vpravo)

Porovnání dvou diagramů

Výstupem porovnání diagramů jsou dva nové diagramy, ve kterých program vyznačí kontrastně shodné proměnné a shodné zpětné vazby a uzly a vazby, které není možné spárovat, nechá šedivé.



Obrázek 31 - zvýrazněné uzly a vazby v kopii porovnávaného diagramu

Konstrukce výstupu porovnávání dvou diagramů je inspirována [Bureš 2017] metodou, ve které dva nebo více diagramů, které pojednávají o tomtéž tématu, postupně překresluji do jednoho diagramu. Pokud se některé proměnné a vazby mezi nimi shodují, nechávám je zakreslené původní barvou a ty, co se neshodují, kreslím také, ale nevýraznou barvou.

Závěry

Vytvořil jsem program, který umí v libovolném příčinném smyčkovém diagramu dojít k cíli a identifikovat smyčky, jejich polaritu a umí porovnat smyčky zakreslené a vyřazovací metodou je napárovat na smyčky vypočtené. Program umí zasáhnout do nativního formátu MDL a zvýraznit v něm chyby. Takto upravený soubor lze programem Vensim znovu otevřít a dále jej zpracovávat.

Pozitivní na celé práci hodnotím to, že algoritmus je konečný a vždy dojde k cíli. Poněkud neuspokojující je rychlost algoritmu pro rozsáhlé diagramy (20+ proměnných), kde bezpečné nalezení všech smyček trvá i desítky minut.

Seznam použité literatury

[1] BECK, Mathias and SCHOENENBERGER, Lukas Klaus and SCHENKER-WICKI, Andrea. **How Managers Can Deal with Complex Issues: A Semi-Quantitative Analysis Method of Causal Loop Diagrams Based on Matrices.** UZH Business Working Paper No. 323., 2012. Available at SSRN: https://ssrn.com/abstract=2573718 or http://dx.doi.org/10.2139/ssrn.2573718

[2] BINDER, Thomas, et al. **Developing system dynamics models from causal loop diagrams.** In: Proceedings of the 22nd International Conference of the System Dynamic Society, 2004.

[3] BUREŠ, Vladimír. **Systémové myšlení pro manažery.** Praha: Professional Publishing, 2011. ISBN 978-80-7431-037-9

[4] BUREŠ, Vladimír. **A Method for Simplification of Complex Group Causal Loop Diagrams Based on Endogenisation, Encapsulation and Order-Oriented Reduction.** Systems 5, no. 3: 46, 2017.

[5] OLIVA, R. **Model structure analysis through graph theory: partition heuristics and feedback structure decomposition.** Syst. Dyn. Rev., 20: 313336., 2004. doi: 10.1002/sdr.298

[6] RICHARDSON, George. **Problems with causal-loop diagrams.** System Dynamics Review 2., 2:158-170, 1986. ISSN 0883-7066

[7] VEČERKA, Arnošt. **Grafy a grafové algoritmy.** Přírodovědecká fakulta Unniverzita Palackého, 2007.

[8] YEARWORTH, Mike, WHITE, Leroy, **The uses of qualitative data in multimethodology: Developing causal loop diagrams during the coding process**, European Journal of Operational Research, Volume 231, Issue 1, Pages 151-161, 2013. ISSN 0377-2217, https://doi.org/10.1016/j.ejor.2013.05.002.

Seznam online zdrojů

[9] FINLEY, Darel Rex. **Point-In-Polygon Algorithm — Determining Whether A Point Is Inside A Complex Polygon**. [online]. 2007, [cit. 19.7.2018]. Dostupné z: http://alienryderflex.com/polygon/

[10] HAINES, Eric. **Point in Polygon Strategies.** [online]. 2001, [cit. 19.7.2018]. Dostupné z: http://erich.realtimerendering.com/ptinpoly/

[11] Vensim. **Dokumentace k programu Vensim.** [online]. 2015, [cit. 19.7.2018]. Dostupné z: https://www.vensim.com/documentation/

[12] Jirovský, Lukáš. **Teorie grafů ve výuce na střední škole.** [online]. 2010, [cit. 16.7.2018]. Dostupné z http://teorie-grafu.cz

Zadání práce

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2016/2017

Studijní program: Systémové inženýrství a informatika
Forma: Kombinovaná
Obor/komb.: Informační management (im3-k)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Petržilka Tomáš	Prorektorská 671/4, Praha - Malešice	I1500314

TÉMA ČESKY:

Analýza příčinných smyčkových diagramů

TÉMA ANGLICKY:

Analysis of causal-loop diagrams

VEDOUCÍ PRÁCE:

doc. Ing. Vladimír Bureš, Ph.D., MBA - KIT

ZÁSADY PRO VYPRACOVÁNÍ:

Vytvořit postup pro analýzu CLD diagramů a jeho ověření.

OSNOVA
1. Úvod
2. Cíl práce
3. Analytická část
3.a. Struktura diagramu
3.b. Uzly
3.c. Vazby
3.d. Kauzální smyčky
3.e. Algoritmus validace
4. Návrhová část
4.a. Rozšíření programu Vensim
4.b. Rozbor datového formátu MDL
4.c. Výběr vhodné varianty
5. Praktická část
5.a. Identifikace prvků v diagramu
5.b. Programování algoritmu
5.c. Prezentace výstupu
6. Ověřovací metodika
6.a. Generování chyby do diagramu
6.b. Ověření chyby v programu Vensim
6.c. Nalezení chyby v modifikovaném diagramu
7. Závěry

SEZNAM DOPORUČENÉ LITERATURY:

1. BECK, Mathias and SCHOENENBERGER, Lukas Klaus and SCHENKER-WICKI, Andrea, (2012), How Managers Can Deal with Complex Issues: A Semi-Quantitative Analysis Method of Causal Loop Diagrams Based on Matrices. UZH Business Working Paper No. 323. Available at SSRN: https://ssrn.com/abstract=2573718 or http://dx.doi.org/10.2139/ssrn.2573718
2. BINDER, Thomas, et al., (2004), Developing system dynamics models from causal loop diagrams. In: Proceedings of the 22nd International Conference of the System Dynamic Society.
3. BUREŠ, Vladimír, (2011), Systémové myšlení pro manažery, Praha, Profesional Publishing, ISBN 978-80-7431-037-9.
4. BUREŠ, Vladimír, (2017), "A Method for Simplification of Complex Group Causal Loop Diagrams Based on Endogenisation, Encapsulation and Order-Oriented Reduction." Systems 5, no. 3: 46.
5. BURNS, James R.; MUSA, Philip. (2001), Structural validation of causal loop diagrams. In: Proceedings of the Atlanta SD Conference: July 2001; Atlanta.
6. JAMIESON, D.W. , BARNETT R.C. & A.F. BUONO (Eds.), (2016), Consultation for organizational change revisited (Research in Management Consulting Vol. 23), pp. 231-254. Charlotte, NC: Information Age Publishing.
7. LIU, S., TRIANTIS, K. P. AND SARANGI, S. (2011), Representing qualitative variables and their interactions with fuzzy

logic in system dynamics modeling. Syst. Res., 28: 245263. doi:10.1002/sres.1064
8. OLIVA, R. (2004), Model structure analysis through graph theory: partition heuristics and feedback structure decomposition. Syst. Dyn. Rev., 20: 313336. doi: 10.1002/sdr.298
9. PANKAJ, KIRAN SETH, SUSHIL. (1994), A fuzzy set theoretic approach to qualitative analysis of causal loops in system dynamics, European Journal of Operational Research, Volume 78, Issue 3, Pages 380-393, ISSN 0377-2217, https://doi.org/10.1016/0377-2217(94)90047-7.
10. SCHAFFERNICHT, M., (2010), Causal loop diagrams between structure and behaviour: A critical analysis of the relationship between polarity, behaviour and events. Syst. Res., 27: 653666. doi:10.1002/sres.1018
11. Vensim help [online]. Ventana systems, (2017) [cit 31.1.2018]. Dostupné z http://vensim.com/documentation/
12. YEARWORTH, Mike, WHITE, Leroy, (2013), The uses of qualitative data in multimethodology: Developing causal loop diagrams during the coding process, European Journal of Operational Research, Volume 231, Issue 1, Pages 151-161, ISSN 0377-2217, https://doi.org/10.1016/j.ejor.2013.05.002.

Podpis studenta: ..... **Datum:** .....

Podpis vedoucího práce: ..... **Datum:** .....