Univerzita Pardubice
Fakulta ekonomicko-správní
Ústav systémového inženýrství a informatiky

Analýza komplexity v informačních systémech

Bc. Vlastimil Pařízek

Diplomová práce
2018

Univerzita Pardubice
Fakulta ekonomicko-správní
Akademický rok: **2017/2018**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Vlastimil Pařízek**
Osobní číslo:	**E16678**
Studijní program:	**N6209 Systémové inženýrství a informatika**
Studijní obor:	**Informatika ve veřejné správě**
Název tématu:	**Analýza komplexity v informačních systémech**
Zadávající katedra:	**Ústav systémového inženýrství a informatiky**

Zásady pro vypracování:

Cílem práce je sumarizovat výhody a nevýhody vysoké a nízké komplexity na různých úrovních abstrakce (dimenzí) v rámci všech fází životního cyklu informačního systému.

Osnova práce:
- Základy a tvorba informačních systémů
- Komplexita informačních systémů
- Postupy a vyhodnocenení analýzy komplexity v jednotlivých úrovních abstrakce IS
- Analýza vybraného infomačního systému
- Hodnocení vybraného systému
- Závěr a zhodnocení práce.

Rozsah grafických prací:

Rozsah pracovní zprávy: **cca. 60 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

BASL, J. Podnikové informační systémy: podnik v informační společnosti. 2., výrazně přeprac. a rozš. vyd. Praha: Grada, 2008. Management v informační společnosti. ISBN 8024722798.
BRUCKNER, T. Tvorba informačních systémů: principy, metodiky, architektury. Praha: Grada, 2012. Management v informační společnosti. ISBN 978-80-247-4153-6.
Xia, W., & Lee, G. (2003). Complexity of Information Systems Development Projects: Conceptualization and Measurement Development. J. Manage. Inf. Syst., 22(1), pp. 45-83.
Subramanian, N. (2011, June). Evaluating Complexity of Information System Architecture Using Fractals. In International Conference on Advanced Information Systems Engineering (pp. 308-317). Springer, Berlin, Heidelberg.

Vedoucí diplomové práce: **Ing. Martin Ibl, Ph.D.**
Ústav systémového inženýrství a informatiky

Datum zadání diplomové práce: **1. září 2017**

Termín odevzdání diplomové práce: **30. dubna 2018**

doc. Ing. Romana Provazníková, Ph.D.
děkanka

L.S.

doc. Ing. Pavel Petr, Ph.D
vedoucí ústavu

V Pardubicích dne 1. září 2017

PROHLÁŠENÍ

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako Školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47 b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 30. 4. 2018 Bc. Vlastimil Pařízek

PODĚKOVÁNÍ:

Tímto bych rád poděkoval svému vedoucímu práce Ing. Martinu Iblovi, Ph.D za jeho odbornou pomoc, cenné rady a poskytnuté materiály, které mi pomohly při zpracování diplomové práce. Dále děkuji své rodině za podporu, které se mi dostávalo po celou dobu studia.

ANOTACE

Diplomová práce se věnuje analýze komplexity v informačních systémech, její příčině a důsledkům pro uživatele a organizace. V práci je vymezen pojem informační systém a jeho životní cyklus. Dále je zde uveden způsob měření komplexity ve všech fázích životního cyklu a všech obsahových dimenzích systému. Definovaný přístup k měření komplexity je následně použit na zhodnocení složitosti na reálně implementovaném systému v soukromé společnosti. Na závěr jsou zhodnoceny získané výsledky analýzy i výstupy z celé práce.

KLÍČOVÁ SLOVA

komplexita, informační systém, životní cyklus, analýza komplexity, MMNDIS

TITLE

Analysis of complexity in information systems

ANNOTATION

The Master Thesis deals with the analysis of complexity in information systems, its cause and consequences for users and organizations. The concept of information system and its life cycle is defined in the thesis. In the work is also presented a way of measuring complexity in all phases of the life cycle and all content dimensions of the system. A defined approach to measuring complexity is then used to evaluate the complexity of a system implemented in a private corporation. Finally, the obtained results of the analysis and outputs from the whole work are evaluated.

KEYWORDS

complexity, information system, life cycle, analysis of complexity, MMNDIS

OBSAH

SEZNAM TABULEK

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

AM	Asset Management
BPMN	Business Process Model and Notation
CMDB	Configuration Management Database
CO	Controlling
DSS	Decision Support Systém
EIS	Executive Information System
ES	Expert System
ERP	Enterprise Resource Planning
FI	Financial Accounting
GUI	Graphical User Interface
HR	Human Resources
ICT	Information and Communication Technologies
IS	Information Systém
MM	Material Management
MMNDIS	Multidimensional Management and Development of Information Systems
MIS	Management Information System
PM	Plant Maintenance
PoZ	Péče o Zařízení
SD	Sales & Distribution
SIS	Strategic Information System
SLOC	Source Lines of Code
TPS	Transaction Processing System
UML	Unified Modeling Language

ÚVOD

Komplexita nebo též složitost je pojem dnes často používán v odborných článcích, metodikách a standardech požívaných k řízení informatiky. Právě komplexitou v informačních systémech se zabývá tato práce. V rámci této diplomové práce je na komplexitu v informačních systémech pohlíženo jako na vlastnost systému, která ovlivňuje jeho přehlednost, obtížnost implementace, modifikovatelnost, pochopitelnost nebo použitelnost. První kapitola je věnována vymezení pojmu informační systém a detailně se bude věnovat jeho životnímu cyklu, a to zejména z důvodu pochopení jednotlivých fází vývoje tak, aby v nich bylo v další části práce možno hodnotit komplexitu.

Cílem práce je sumarizovat výhody a nevýhody vysoké a nízké komplexity na různých úrovních abstrakce (dimenzí) v rámci všech fází životního cyklu informačního systému (dále jen IS). Tomuto se bude věnovat kapitola 2, která se bude zabývat komplexitou a jejímu vlivu na jednotlivé fáze abstrakce v rámci všech fází životního cyklu. V této kapitole bude také uveden způsob kvantifikace komplexity během vývoje IS i v rámci jeho provozu s využitím vybrané metodiky řízení informačních systémů.

V poslední kapitole práce budou na vybraném reálném informačním systému použity postupy definované v kapitole 2. Na implementovaném systému SAP bude hodnocena jeho komplexita zejména z pohledu uživatele a také bude nastíněn multiplikativní efekt komplexity tohoto systému při provádění jeho úprav. Budou také vytipovány možné příčiny nadbytečné komplexity.

1. INFORMAČNÍ SYSTÉM A JEHO ŽIVOTNÍ CYKLUS

Tato kapitola bude věnována informačním systémům. Nejprve bude vysvětlen pojem informační systém a dále budou popsány základní druhy infomačních systémů dle jejich postavení v systému řízení podniku. Následující část kapitoly bude zaměřena na životní cyklus informačního systému a různé přístupy k jeho vývoji.

1.1. Informační systém

Jednoznačně definovat pojem informační systém není snadné, a proto neexistuje žádná jednoznačná definice, protože každý uživatel nebo tvůrce informačního systému používá různé terminologie a zdůrazňuje jiné aspekty. Lze však tvrdit, že informační systém lze popsat jako vzájemné propojení uživatelů, hardwaru, softwaru, informací a procesů, které s těmito informacemi pracují. Pod pojmem procesy rozumíme funkce, které zpracovávají informace, které do systému vstupují a transformují je na informace, které ze systému vystupují. Procesy se tedy rozumí funkce zabezpečující sběr, přenos, uložení, zpracování a distribuci informací.

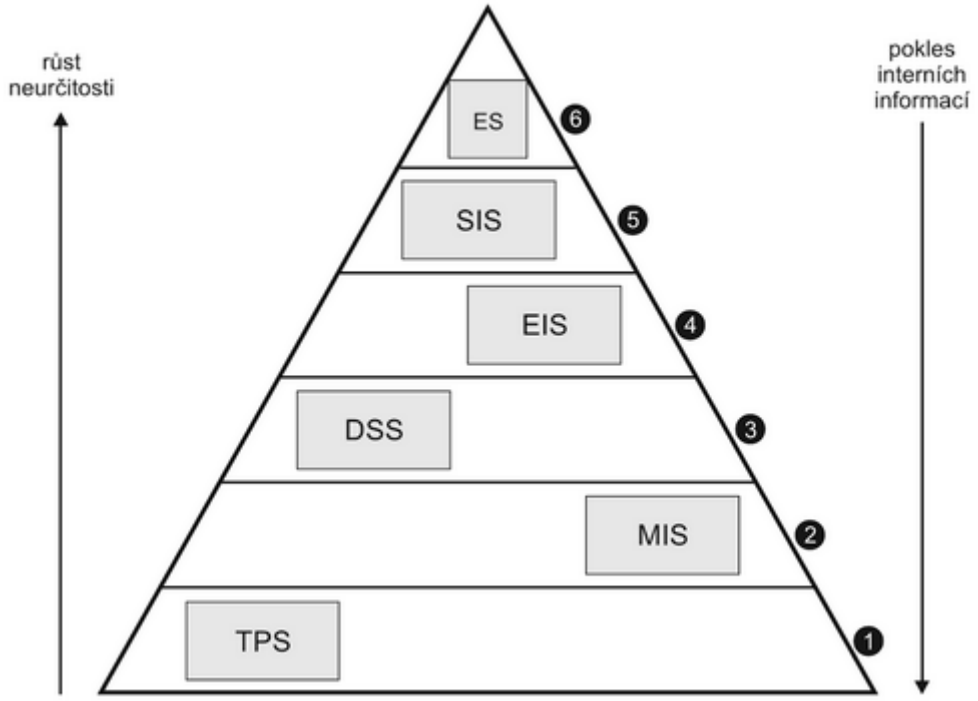
Jako každý systém má i informační své okolí. Okolí informačního systému tvoří veškeré objekty, které nějak ovlivňují samotný systém, a také objekty, které jsou systémem ovlivněny.

Celkově lze říci, že IS je softwarové vybavení, které je schopné na základě zpracovávaných informací řídit procesy. Dále je schopen tyto informace poskytovat řídícím pracovníkům tak, aby jim byl nápomocný zejména při plánování, koordinaci a kontrole veškerých procesů, a to jak v soukromém, tak veřejném sektoru. [28]

1.2. Druhy IS

Informační systémy je možno rozdělit do kategorií dle různých hledisek. Hlavním kritériem je účel využití systému. Dále lze systémy rozlišit dle obsahu, velikosti, strukturální složitosti, počtu a typu uživatelů a územního rozsahu. Pro volbu metod a přístupů k řízení IS je ale rozhodující především vztah IS k účelu jeho využití ve vztahu k systému řízení podniku. Z hlediska postavení IS v systému řízení rozlišujeme především to, v jakém stupni informační pyramidy se nachází. Informační pyramida je na Obrázek 1. Čím výše se systém v pyramidě nachází, tím více přibývá neurčitost v požadavcích na informační systém, ale také ubývá objem

interních informací. Díky tomuto přibývá potřeba externích informací z podstatného okolí systému. [7]



Obrázek 1: Informační pyramida

Zdroj: [13]

1.2.1. Transakční systémy (TPS)

Transakční systémy mají za cíl mechanizovat typické agendové úlohy, jako jsou mzdy, fakturace nebo skladové hospodářství. Uživatelé transakčních systémů jsou vysoce kvalifikovaní pracovníci schopní provádět samostatná rozhodnutí v zájmu podnikových cílů. Uživatelské rozhraní těchto systému je poměrně komplexní proto je kladen důraz na kvalitu uživatelů, kteří se systémem denně pracují. [9]

1.2.2. Systémy pro řízení (MIS)

Systémy vycházející z účetních a ekonomických systémů. Jejich hlavní funkcí je zpřístupnění různých přehledů nebo tvorba sestav, čímž usnadňují práci řídícím pracovníkům, hlavně při kontrole výkonnosti. Typické jsou detailní přehledy o provozu některých dílen, provozů, závodů i celých podniků. [9]

1.2.3. Systémy pro podporu rozhodování (DSS)

Systémy podpory rozhodování jsou informační systémy, které podporují podnikové nebo organizační rozhodovací činnosti. Tyto systémy mohou být využity na všech úrovních řízení podniku, ale je zpravidla používán středním a vyšším managementem. Pomáhají lidem rozhodovat o problémech, které se mohou rychle měnit a nejsou předem jednoznačně specifikovány.

Pod tyto systémy někteří autoři zahrnují všechny systémy, které by mohly podporovat rozhodování. Správná definice je dle [27] takováto:

- DSS májí tendenci být zaměřeny na méně strukturovaný, nespecifikovaný problém, který obvykle čelí manažeři vyšší úrovně, pokouší se kombinovat použití modelů nebo analytických technik s tradičními datovými funkcemi a funkcemi vyhledávání;
- DSS obsahují funkce, které jsou snadno použitelné i pro uživatele, který nemá počítačové vzdělání;
- DSS se zaměřují na flexibilitu a přizpůsobivost ke změnám v přístupu k rozhodování uživatele. Některé DSS zahrnují také znalostní systémy.

Správně navržený systém DSS je interaktivní softwarový prostředek určený k tomu, aby pomáhal rozhodujícím osobám sestavovat užitečné informace z kombinace prvotních dat, dokumentů a osobních znalostí nebo obchodních modelů k identifikaci a řešení problémů a přijímání rozhodnutí.

1.2.4. Informační systémy pro vrcholové řízení (EIS)

EIS jsou aplikace účelově orientované na potřeby podpory vedení podniků a institucí. Využívají všech dostupných informačních zdrojů vytvářených na nižších úrovních informačního systému, tj. úlohami transakčního charakteru, úlohami pro taktické a operativní řízení a úlohami pro podporu rozhodování. [29]

1.2.5. Strategické informační systémy (SIS)

SIS se snaží se o zvýšení konkurenceschopnosti podniku. Jsou přímo spjaty s výrobou nebo výrobkem. V průmyslové oblasti čerpají informace třeba z automatizovaných výrobních linek a v obchodu například z elektronické pošty.

1.2.6. Prognostické expertní systémy (ES)

Tyto systémy jsou vytvořeny z nástrojů, které dovolují provádět analýzy, které odpovídají na otázky typu „CO KDYŽ?“ a vytvářet prognózy. Do této kategorie spadají také expertní systémy (ES). Tyto systémy jsou založeny na množině pravidel, která pomáhají nepříliš zkušenému a znalému pracovníkovi řešit úlohy často diagnostického charakteru. Hlavním cílem zavedení expertního systému je poskytnout znalosti, které vlastní pouze několik expertů v oboru, více pracovníkům. Tyto systémy využívají technologie z oblasti umělé inteligence v podobě soustav pravidel produkčního typu uložených v tzv. bázi znalostí (soustava IF-THEN pravidel).

1.3. Životní cyklus informačního systému

Před vznikem každého informačního systému je vlastní rozhodnutí o tom, zda a jaký systém bude pořízen a jaké procesy bude pokrývat. Řízení vývoje nového informačního systému je velmi náročnou činností. Životní cyklus, jak již název napovídá, postihuje celý „život“ systému od bodu kdy je rozhodnuto o jeho potřebě, až do doby, kdy je jeho provoz ukončen. Rozdělení na jednotlivé fáze cyklu je možné provézt několika způsoby. Nejčastější rozdělení je na 4 etapy: **plánování, návrh, zavádění** a následně **provoz a údržba**.

1.3.1. Fáze plánování

Jak bylo uvedeno na začátku této kapitoly, je vždy na začátku každého informačního systému rozhodnutí o jeho pořízení. V rámci tohoto musí být známy jednotlivé požadavky všech uživatelů a zejména cíle, kterých je pomocí realizace projektu vývoje IS třeba dosáhnout. V tomto období se shromažďují všechny požadavky na systém. Je vytvořen hrubý popis funkčnosti systému a jeho využití v rámci procesů v organizaci. V této části projektu je také odhadnuta doba realizace vývoje IS a přibližné náklady. Výsledkem snažení v této fázi je dokument specifikující účel systému, identifikující uživatele a jejich zásadní požadavky. Dále obsahuje: definice částí systému a návrh jejich řešení, seznamy událostí, odhady datové základny, hardwarové a softwarové zajištění.

Kromě funkčních požadavků jsou v této části také stanoveny požadavky na výkonnost systému. Jsou stanoveny a číselně vyjádřeny změřitelné požadavky na výkonnostní aspekty systému. Mezi tyto požadavky patří jak **statické**, jako je počet podporovaných uživatelů, tak i **dynamické** mezi, které patří počet transakcí zpracovaných za jednotku času nebo doba odezvy systému při různých zátěžích. Číselné a měřitelné vyjádření je nutné pro ověření, zda

implementovaný systém tyto požadavky naplňuje. Nutné je se vyvarovat subjektivním hodnocením, jako je „málo“ nebo „dlouho“.

Dále se v rámci této fáze specifikují další požadavky na vlastnosti systému, jako je:

- **Dostupnost –** udává úroveň zaručené dostupnosti systému, popisuje kontrolní body a schopnost systému vzpamatovat se po výpadku.
- **Bezpečnost** – zahrnuje ochranu systému a dat proti zneužití.
- **Udržovatelnost** – je určeno, jak má být systém členěn s výhledem na budoucí údržbu.
- **Přenositelnost** – říká, na jakých platformách bude možné systém provozovat včetně možných omezení.
- **Snadnost užívání** – definuje nároky na uživatele.

I pro tyto požadavky platí, že musí být **měřitelné**, aby bylo možné jejich plnění ověřit.

Dalším aspektem, kterým je třeba v této fázi definovat je rozhraní. Rozlišuje se zde rozhraní **uživatelské**, kde jsou definovány formáty a vzhled obrazovek a dialogů, časování a formáty hlášení, dále pak **hardwarové** určující požadavky pro zajištěnou plnou funkčnost systému**, softwarové,** vztahující se k programům použitým při vývoji a k dalším programům používaných v organizaci, **komunikační** definující komunikaci v síti v rámci organizace.

Tato fáze vývoje informačního systému je velmi obsáhlá a složitá, proto je nutné zvolit vhodnou členitost, aby byly požadavky přehledné a dobře čitelné všem členům vedení, kteří provádí závěrečná rozhodnutí. Na základě informací z této fáze rozhodne vedení organizace o realizaci projektu. [2]

1.3.2. Fáze návrhu

Na základě získaných informací z předchozí fáze je vytvořen konceptuální model popisující datovou i funkční podstatu zachycené části podniku, která má být zanesena do informačního systému. Jsou vytvořeny logické a datové modely, určeny podmínky zavádění v organizaci, záruční servis a podmínky celkového zavedení informačního systému.

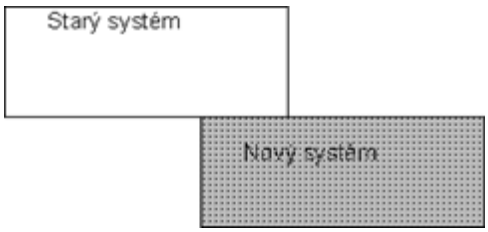
Po tomto je možno přistoupit k vlastnímu řešení systému zvoleným softwarem. Tato část životního cyklu IS zahrnuje vlastní programování, kdy jsou implementovány datové a funkční vlastnosti řešené oblasti organizace. Hlavními členy realizačního týmu jsou vybraní softwaroví experti a analytik, který je odpovědný za správnost řešení a splnění požadovaných cílů. Na základě získaných faktů z fyzického návrhu z předchozí etapy se definují vstupy a výstupy jednotlivých operací a určí se způsob jejich modifikace. Naprogramují se veškeré funkce a na testovacích datech se prověří jejich správnost, zda byly vytvořeny tak, aby splňovali stanovené požadavky. Vzhledem k tomu, že jsou použita testovací data mimo reálnou praxi je zde prostor na odhalení možných chyb, bez jakýchkoliv následků na organizaci. Toto je hlavním důvodem, proč je nutno této fázi věnovat dostatečné množství času a úsilí.

1.3.3. Fáze zavádění systému

Kompletně otestovaný systém je třeba v dalším kroku nasadit do provozu v daném prostředí, pro které byl vytvořen. Provádí se jeho instalace, transformace původní datové základny a probíhá školení uživatelů. Při školení se zpravidla začíná s vedoucími pracovníky a dále se s programem seznamují zaměstnanci v provozu. Postupů pro zavádění IS do ostrého provozu je několik. V praxi se nejběžněji užívají následující:

- **Souběžné zavádění**

Informační systém je zaveden souběžně se starým systém. Tento postup je vhodné použít při zavádění jednodušších IS, které jsou jednoduché na ovládání a nejsou proto nutná zdlouhavá školení uživatelů. Uživatelé souběžně obsluhují nový i starý systém po uplynutí dané doby je starý systém odstraněn a uživatel již využívá pouze nový. Princip této varianty zavádění je na Obrázek 2.

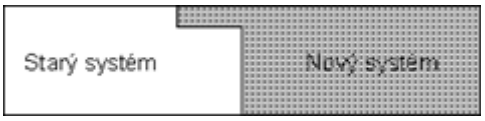


Obrázek 2: Souběžné zavádění

Zdroj: [14]

- **Pilotní zavádění**

Informační systém je zaveden na jednom pracovišti. Na tomto pracovišti se zacvičí i uživatelé z ostatních pracovišť, kam je systém zaveden až po tom, co je pilotní část systému vyzkoušena a uživatelé zaškoleni. Tento způsob je vhodný pro zavádění kvalitativně odlišných IS, které vyžadují rozsáhlé testování v provozních podmínkách. Toto pilotní zavádění umožňuje postupnou transformaci dat z předchozích IS. Schématický je tato varianta znázorněna na Obrázek 3

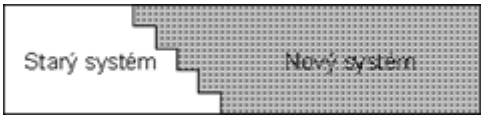


Obrázek 3: Pilotní zavádění

Zdroj: [14]

- **Postupné zavádění**

Využívá se zejména u velice složitých systémů se složitými vnitřními vazbami. Nejprve se zavádějí primární části systému, na kterých ostatní části závisí, po jejich ověření v provozu jsou zavedeny i ostatní části sytému. Obrázek 4 ilustruje postup této metody.

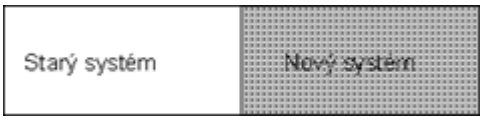


Obrázek 4: Postupné zavádění

Zdroj: [14]

- **Nárazová strategie zavádění**

Na Obrázek 5 je tento způsob zavádění znázorněn. Při tomto způsobu zavádění je starý systém ihned nahrazen novým. Činnost starého systému je ukončena nárazově. Tento postup je velmi riskantní, používá se jen tam, kde není jakýkoliv souběh více systémů možný.



Obrázek 5 :Nárazová strategie

Zdroj: [14]

V praxi však nastává nutnost kombinovat jednotlivé postupy. Nejčastější je kombinace postupu nárazového a postupného. [20]

Po zavedení systému následuje jeho zkušební provoz, kdy je systém plně využíván v provozu a poskytovatel (dodavatel) systému je povinen zajistit okamžitý servis a odstranit chyby zjištěné během provozu v rámci reklamace.

1.3.4. Fáze provozu a údržby

Nejdelší etapa životního cyklu systému. Systém je plně využíván pro potřeby organizace. Do této etapy také spadá údržba systému, tedy zajištění správného provozu, úprava parametrů aplikací nebo změny některých programů tak, aby splňovaly nové požadavky uživatelů. Dříve či později, však dojde ke změně požadavků na systém a je třeba provézt tzv. **reengineering**, kdy nastává přehodnocování požadavků na systém. V případě, že nové požadavky není systém schopen splnit pomocí dílčí úpravy, je nutno se opět v rámci životního cyklu vrátit na jeho začátek.

1.4. Další modely životních cykly IS

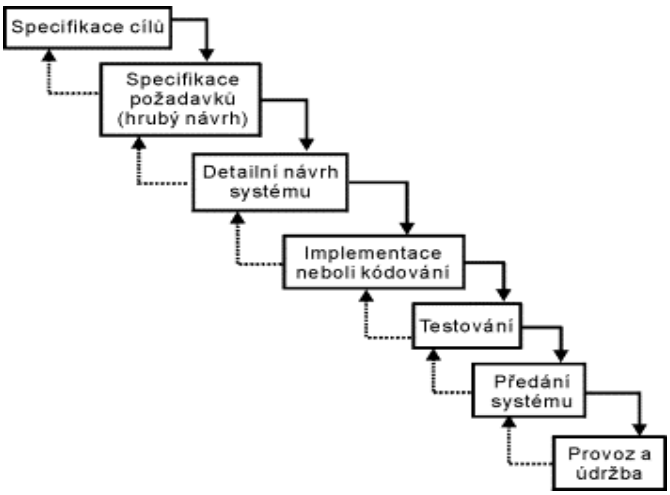
Při vývoji a správě informačních systémů se využívá více různých modelů životního cyklu. V následující části práce jsou uvedeny a vysvětleny ty nejčastější.

1.4.1. Model vodopád

Jedná se klasický model používaný již v 70. letech 19. století je. Vznikl z důvodu nutnosti zavedení jednotného postupu do vývoje systémů, který umožní řešení komplexnějších problémů díky hierarchické dekompozici a snížení množství chyb precizní kontrolou všech výstupů jednotlivých etap.

Základní charakteristikou tohoto modelu je posloupnost na sebe navazujících etap, které se vzájemně neprotínají. Výstup z jedné etapy je vstupem do etapy následující. Jednotlivé etapy jsou prováděny dle plánu a po skončení etapy se k ní již během cyklu nevrací.

Výhodou tohoto modelu je jeho rychlost a cenová nenáročnost, za předpokladu, že se nevyskytnou problémy. Vhodný pro řešení situace, kdy je znám problém i způsob jeho řešení. Tento typ modelu není příliš vhodný pro řešení složitých systémů, kde není přesně definovaný problém a lze tedy obtížně řešit situaci v krocích definovaných modelem na Obrázek 6.



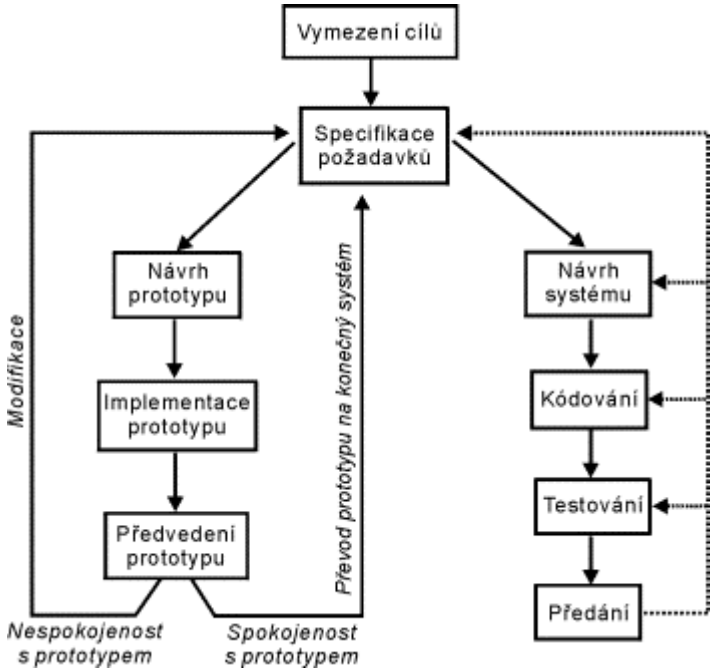
Obrázek 6: Vodopádový model životního cyklu

Zdroj: [2]

Z Obrázek 6 vyplývá, že výsledek projektu zjistíme až po poslední fázi, ve chvíli, kdy je systém předán zákazníkovi a chyby, které jsou objeveny až nyní je velmi složité a nákladné odstranit. V praxi je problém ve velké časové prodlevě mezi objednávkou systému a první verzí předané zákazníkovi, který je již často netrpělivý a další spolupráce s ním je obtížnější. Vodopádový model lze považovat za základní, který je v praxi použitelný hlavně pro malé aplikace, ale i pro velké systémy je jeho metodika lepší než náhodné a živelné řešení problému.

1.4.2. Prototypový model

Model vychází z myšlenky, že zákazník zjistí chyby ve specifikaci problému až v době, kdy mu je předložen hotový systém. Model se začal využívat v 80.letech 19. století a jeho hlavním cílem je předložit zákazníkovi první verzi systému, která buď zjednodušeně obsahuje celý řešený prostor, nebo postihuje kompletně pouze část řešeného problému. Prototyp je realizován v co nejkratším čase a v takové funkčnosti, která prezentuje veškerá vnější rozhraní a umožňuje zákazníkovi reagovat na výsledky. Následné změny se provádí na základě připomínek uživatelů, které upřesňují původní požadavky. Postupně je prototyp vylepšován až do doby, dokud není zákazník zcela spokojen, čímž vznikne plnohodnotný požadovaný systém. Model je znázorněn na Obrázek 7.



Obrázek 7: Prototypový model životního cyklu

Zdroj: [2]

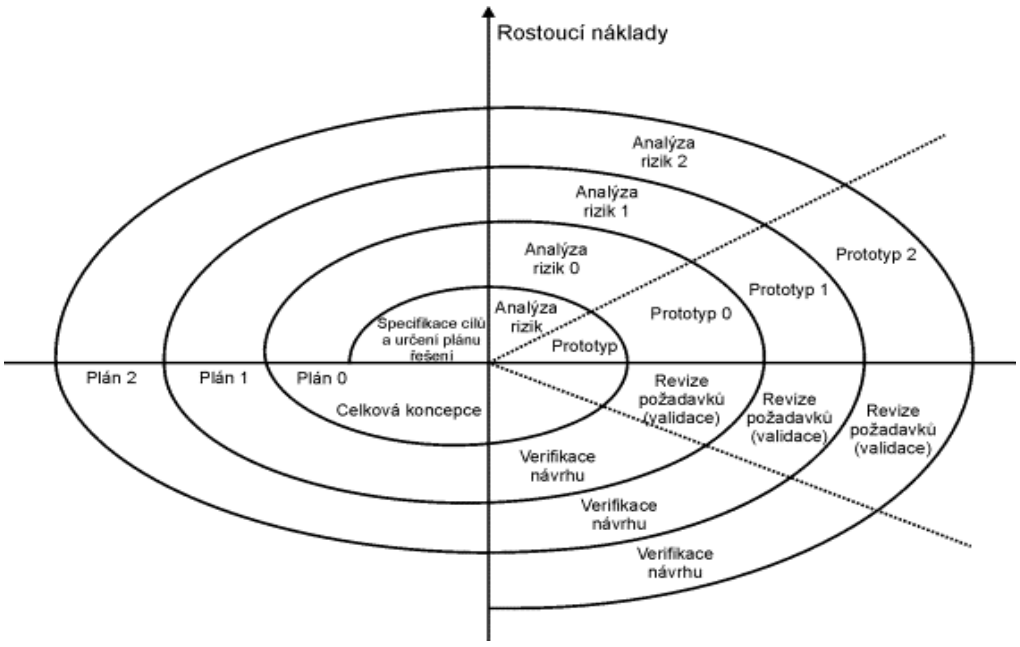
Výhodou tohoto modelu je to, že poskytuje místo pro reakci na změny ze strany uživatelů, na které je možno relativně rychle reagovat. Problém se splněním časového rámce stanoveného ve smlouvě se zákazníkem může nastat v případě, že z důvodu nevystihnutí připomínek uživatelů je prototyp nutno upravovat ve velkém množství cyklů.

1.4.3. Model spirála

Tvůrce tohoto modelu byl v roce 1988 B. W. Boehm. Model kombinuje prototypový přístup a analýzu rizik. Jednotlivé vývojové kroky se neustále opakují, tak že v každém dalším kroku se na již ověřenou část systému přibalují části na vyšší úrovni. Každý krok se skládá z následujících částí:

- Specifikace cílů a určení plánu řešení.
- Vyhodnocení možných řešení a analýza rizik řešení.
- Vývoj prototypu dané úrovně a jeho předvedení a vyhodnocení.
- Revize požadavků a testování správnosti prototypu.
- Ověření výstupu daného kroku je-li v souladu se zjištěnými požadavky.

Model využívá ověřené kroky vývoje a s využitím analýzy rizik předchází možným chybám. Požadavky zákazníků jsou konzultovány v jednotlivých krocích, což umožňuje systém postupně modifikovat dle přání uživatele. Uživatel má možnost ovlivnit již vývoj prvního prototypu. Nevýhodami je zejména časová náročnost spolupráce pro zákazníka, špatně předvídatelné náklady a celková časová náročnost. Je nutné klást velký důraz na správnost analýzy rizika, kdy špatně ohodnocené riziko může mít velký vliv na celý projekt. Schéma modelu vyjadřuje Obrázek 8.



Obrázek 8: Spirálový model životního cyklu

Zdroj: [2]

Zvolení správného modelu životního cyklu má velký dopad na výslednou komplexitu systému. Ovšem při zvolení jakéhokoliv modelu je třeba kompletně a detailně zmapovat dané podnikové procesy. Při tvorbě informačního systému jsou ve fázi návrhu vytvářeny modely, které postihují důležité prvky systému a jejich vazby pro pochopení fungování procesu. Pomocí modelů je vystižena jak funkční, tak datová složka systému. Při modelování se používá tzv. **princip tří architektur**, který definuje způsob použití abstrakce ve vývoji informačního systému po jednotlivých vrstvách. Jednotlivé vrstvy se zaměřují na tři hlavní aspekty vyvíjeného systému: obsah, technologii a implementační specifika. Ze specifikace obsahu vyplývají možnosti technologického řešení a konkrétní použitá technologie určuje implementační možnosti. Návrh probíhá ve třech úrovních:

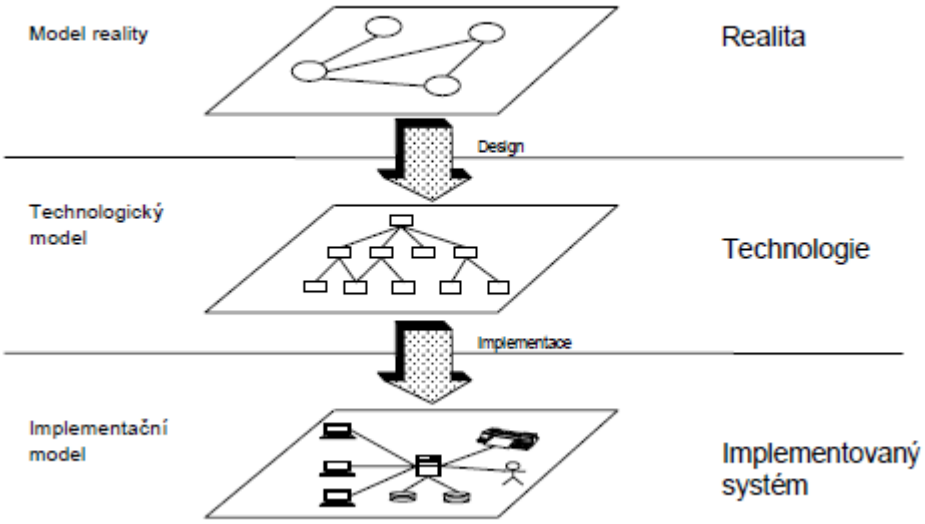
- Konceptuální – obecný obsahový model, nezatížený technologickou koncepcí řešení ani implementačními specifiky. Určuje, **CO** je obsahem systému.
- Technologická – model systému zohledňující technologickou koncepci řešení, koncepci organizace dat a jejich zpracování. Určuje, **JAK** je obsah systému v dané technologii realizován.
- Implementační – model systému zohledňující implementační specifika použitého vývojového prostředí (databázový systém, programovací jazyk…)

Návaznost jednotlivých vrstev je na Obrázek 9 [25]

Princip hierarchické abstrakce

Hierarchická abstrakce je prostředkem rozkladu prvků vyvíjeného systému do detailní úrovně pohledu. Pojmy vyšší úrovně sestávají z pojmů nižší úrovně. Na každé úrovni jsou popsány její jednotlivé prvky a vazby mezi nimi. Prvky vyšších úrovní popisu jsou abstraktními prvky a mohou být popsány na nižší úrovni pomocí prvků, z nichž každý jeden může být jak abstraktním, tak konkrétním elementárním prvkem. Existují dva základní typy hierarchické abstrakce:

- **Abstrakce část-celek (kolektivizace, agregace).** Používá se ve funkčním modelu, kde se systém dělí na subsystémy, části subsystému atd. Vyšší celek je zcela definován jako souhrn svých částí.
- **Abstrakce podtyp-obecný typ (generalizace).** Používá se v datovém modelu, kde je možné uvažovat o jednotlivých specifických variantách nadřazeného pojmu (entity, objektu). Na rozdíl od agregace není nadřazený celek definován jako souhrn podřízených částí, ale jako nositel jejich společných vlastností (atributů).



Obrázek 9: Vrstvy vývoje IS

Zdroj: [24]

2. KOMPLEXITA INFORMAČNÍCH SYSTÉMŮ

Tato kapitola se věnuje komplexitě informačních systémů. Nejdříve je třeba pojem komplexita definovat. Po objasnění tohoto pojmu ve vztahu k infomačnímu systému je dále kapitola věnována možnosti měření její míry v rámci životního cyklu IS. Pro tento účel je využita metodika vývoje informačního systému MMNDIS. V závěru kapitoly je uveden přehled výhod a nevýhod vysoké a nízké komplexity v jednotlivých fázích životního cyklu IS.

2.1. Komplexita

Komplexitou je možno zjednodušeně nazvat komplikovanost či složitost nějakého systému. Komplexní systém jako celek vykazuje jednu nebo více charakteristických vlastností, které však nejsou jasně viditelné z vlastností jednotlivých částí systému. Teorie komplexity vychází z postulátu „celek je víc než suma jeho částí“. U komplexních systémů se vyskytuje tzv. emergence, což je vlastnost komplexních systémů, kdy v celkovém chování lze odhalit komplexní vzory vznikající z jednodušších pravidel. Pod pojmem vzor je myšleno zejména podobnost v chování. U komplexního systému nejde o počet prvků, ale o složitost jejich propojení, hustotu sítě. Důsledkem rostoucí komplexity je větší složitost systému, která způsobuje například jeho horší čitelnost, srozumitelnost či použitelnost. [16]

Komplexita se vztahuje k implicitní povaze systému, která se dotýká jak vlastností interagujících komponent, tak povahy jejich interakce. Komplexita systému zahrnuje aspekty jako je neurčitost, fluktuace, singularita, vnitřní dynamika, konektivita a další. Pojem varieta je míra komplexity v systému, která je definovaná jako počet možných stavů.

Druhy rozeznávané komplexity:

- Deskriptivní komplexita – odpovídá tomu, jak se systém jeví pozorovateli, který stojí mimo něj. Nabízí obvykle statické pohledy na systém.
- Přirozená komplexita – je vztažena k vnitřku systému, vystihuje vnitřní podstatu systému a vystihuje i jeho dynamiku.

Ve výše zmíněných druzích komplexity je zakotvena informace nutná k popsání systému, tak i informace k objasnění neurčitosti. V praxi se obě komplexity dostávají do konfliktu. Jestliže chceme omezit jednu z nich, pak druhá pravděpodobně roste nebo v nejlepším případě zůstává stejná. Tato vzájemná výměna je jedním z nejvýznamnějších metodologických východisek systémové vědy.

Aspekty ovlivňující komplexitu každého systému jsou zejména:

• množství prvků a vazeb mezi nimi;

• rekurzivní vztahy (systém nezávisí na jen na vstupních podnětech, ale i na minulém vývoji);

• paralelní procesy a jejich vzájemná synchronizace;

• interakce (vzájemné ovlivňování procesů a přítomnost efektu motýlích křídel)[1];

• neurčitost a fluktuace (všechny systémy udržují určité vzory). [18]

Předmětem této práce je komplexita v kontextu informačních systému. Informační systémy automatizují podnikové procesy, a proto je třeba se zaměřit na komplexitu procesů, které mají být automatizovány, neboť procesy s velkou komplexitou, které je těžké pochopit nelze dobře zavézt do systému a způsobují velké problémy při návrhu systému i jeho implementaci. Každý podnikový proces je sled vzájemně na sebe navazujících činností, které probíhají napříč organizačními jednotkami a reagují na různé podněty z vnitřního i vnějšího prostředí. Proces obstarává transformaci vstupů na výstupy za pomocí nějakých zdrojů. Obecně lze říci, že vždy dochází k nějakému toku činností od jednoho člověka k druhému. [17]

Komplexita podnikových procesů musí být neustále pod kontrolou, aby nedocházelo k jejímu zvyšování v čase, což by mohlo vézt k chybám a ztěžovalo by jejich pochopení. Míru složitosti lze odvodit dle toho, jak obtížné je daný proces pochopit nebo vysvětlit. Komplexita procesů není konstantní. Tak jak se proces v čase vyvíjí, mění se i jeho komplexita.

2.2. Míra komplexity

Určení míry komplexity není jednoduché. Nejzákladnější možnost pro posouzení komplexity je hodnotit systém dle jeho velikosti. Tento přístup však dle definice komplexity uvedené výše není vyhovující, neboť velikost nemusí být přímo spojena se složitostí. Dalším hlediskem může být entropie, kde však musí být zkoumána nějaká forma zprávy a nejvyšší hodnotu má náhodný systém. Lze také měřit algoritmický informační obsah. Tato metoda souvisí s teorií složitosti a slouží pro klasifikaci výpočetních systémů. Zde se rozlišuje algoritmická komplexita, jako nejkratší délka programu popisující nějaký objekt a efektivní komplexita, kterou pro nějakou entitu definuje Murray Gell–Mann jako délku vysoce komprimovaného popisu její pravidelnosti. V této metodě se jedná o snahu oddělit komplexitu a náhodnost. Tento typ komplexity je také označován jako Komodorova-Chaitinova

[1] Efekt motýlích křídel – vyjadřuje citlivou závislost vývoje systému na počátečních podmínkách, jejichž malé změny mohou mít za následek velké variace v delším průběhu

komplexita. Komplexitu lze také měřit pomocí logické hloubky, která určuje, jak je těžké objekt sestrojit a termodynamické hloubky, což je sekvence událostí vedoucí k objektu, včetně zdrojů vyžadovaný ke konstrukci systému. Další metodou je statistická komplexita, která je založena na určení minima informace nutné k předpovědi budoucnosti systému. Dále jsou na měření komplexity používané metody založené na teorii fraktální dimenze či stupně hierarchie. [16]

2.3. Komplexita v IS

Informační systém má za úkol podporovat podnikové procesy. Výše bylo uvedeno, že každý podnikový proces je charakterizován jistou mírou komplexity, tak i jeho podpora v rámci informačního systému nese obdobnou složitost. V dnešní době je tento pojem často uváděn v mnoha metodikách a standardech používaných k řízení informatiky. Pojem komplexita ale bývá zřídka kdy jasně definován. Na složitost informačního systému je možno nahlížet z několika pohledů:

- Jak velké množství dat a informací je zpracováno.
- Jak komplikované procesy jsou systémem řešeny.
- Jak složitá je organizační struktura podniku, který využívá IS.
- Jak je přehledné uživatelské rozhraní.

Pro každý podnik je velmi důležité požívat sofistikovaný SW a HW, neboť oboje má dopady na celou organizaci včetně jejích pracovníků, ale i na ekonomické výsledky. Ve vztahu na velikost organizace je intuitivní, že podnik s velkou a složitou organizační strukturou bude potřebovat také velmi obsáhlý informační systém nebo i vice různých dílčích systémů. Je třeba, aby informační systém dokázal efektivně pokrýt co možná nejvíce požadavků plynoucích ze struktury podniku, ale neobsahoval komplexitu větší, než je rámec těchto požadavků. Ideální informační systém by měl pokrýt pouze ty požadavky, jejichž pokrytí přinese vyšší užitek, než jsou celkové náklady spojené s jejich pokrytím. [10]

Na růst komplexity informačního systému má velký vliv složitá organizační struktura podniku, která je definovaná bez znalosti technologických aspektů. Toto má zásadní vliv na nárůsty komplexity v dalších fázích vývoje a implementace informačního systému, protože s sebou přinese nárůst struktury dat a hrozí jejich redundance. Klade také nároky na design, implementaci a údržbu systému oprávnění pro práci se systémem.

Rostoucí komplexita se nepromítá pouze do oblasti softwaru, ale má velké dopad i na související hardware, kde rostoucí složitost přináší zvyšující se náklady nejen na jeho pořízení, ale hlavně náklady spojené s jeho administrací, provozem a údržbou. Důležitým prvkem každého informačního systému je jeho uživatelské rozhraní, protože má přímý vliv na efektivitu práce uživatele. Složitá a zdlouhavá práce se systémem má neblahý dopad na efektivitu zaměstnanců podniku.

Z výše uvedeného plyne, že komplexitu informačních systémů je nutno řídit, protože v opačném případě roste a s ní rostou podniku náklady a snižuje se efektivita. Již vzniklá velká složitost v informačních systémech se obtížně snižuje. Proto je nutné nejen nedopustit její zvyšování již během návrhu a vývoje informačního systému, ale i během jeho provozu a údržby. Výsledná obsažená optimální komplexita informačního systému musí být vždy ta nejmenší možná, kterou lze dosáhnout. [10]

V současné době je složitost informačních a komunikačních technologie čím dál větším problémem. Stále větší množství funkcí, které informační technologie nabízí, klade také velké nároky na uživatele. Informační technologie ovlivňují běžný život každého člověka v několika směrech. Komplexní systémy s sebou nesou obrovské náklady na jejich nákup, implementaci, poradenství a údržbu ale také představuje riziko pro zajištění kontinuity provozu. Pro zajištění optimální komplexity je nutné při návrhu systému důkladně porozumět podnikové a informační architektuře, ale i rozpoznat mechanismy vedoucí k jejímu vzniku. [11]

2.4. Měření komplexity IS

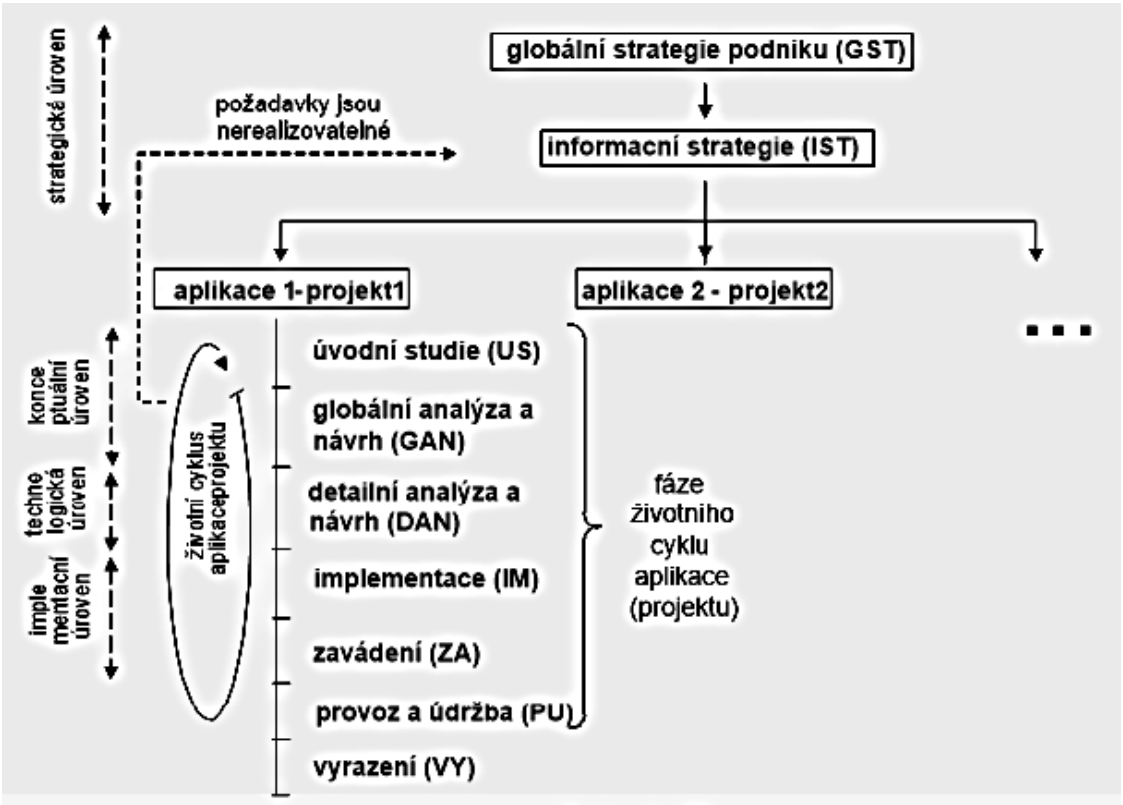
V předchozí podkapitole bylo zmíněno, že komplexitu je obtížné snižovat v již exitujícím složitém systému. Proto je nutné složitost sledovat průběžně v rámci životního cyklu informačního systému. Popis vybraných životních cyklů byl uveden v kapitole 1.3. Pro měření komplexity bude využita metodika MMDIS (Multidimensional Management and Development of Information Systems). Cílem této metodiky je vývoj, údržba a provoz komplexního a integrovaného informačního systému podniku, který optimálně využívá potenciálu dostupných informačních technologií a informatických služeb k maximální podpoře podnikových cílů, přičemž nemusí být postaven na nejnovějších, nejsofistikovanějších technologiích a službách, ale vybírá z nich ty, které mají ekonomický smysl.

Metodika zohledňuje všechny faktory (dimenze), které ovlivňují informační systém v jeho životním cyklu a dá se použít jak pro vývoj nových prvků informačního systému, tak i pro rozvoj již existujícího informačního systému. [19]

V metodě MMDIS jsou rozlišovány tyto dimenze:

- Funkce/procesy;
- Data/informace;
- Organizační a legislativní aspekty;
- Pracovní, sociální a etické aspekty, aspekty lidských zdrojů;
- Software;
- Hardware;
- Uživatelské rozhraní;
- Bezpečnost;
- Ekonomické a finanční aspekty.

Struktura MMDIS je rozdělena na následující části: fázi globální podniková strategie, informační strategie, úvodní studii, globální analýzu a návrh, detailní analýzu a návrh, implementaci, zavádění systému, provoz a údržbu a vyřazení systému. Struktura metodiky MMDIS je znázorněna na Obrázek 10.



Obrázek 10: Struktura metodiky MMDIS

Zdroj: [21]

Při zkoumání komplexity je nutno nějakým způsobem určit její hodnotu pro každou fázi a dimenzi, je tedy třeba obsahové dimenze kvantifikovat. Z tohoto důvodu jsou použity modely, které se v jednotlivých fázích vývoje informačního systému využívají. U každého modelu lze přímo spočítat počet prvků a vzájemných vazeb. Složitost informačního systému je tedy dána součtem složitostí jednotlivých UML modelů v jeho jednotlivých obsahových dimenzích. Struktura používaných UML[2] modelů je na Obrázek 11.

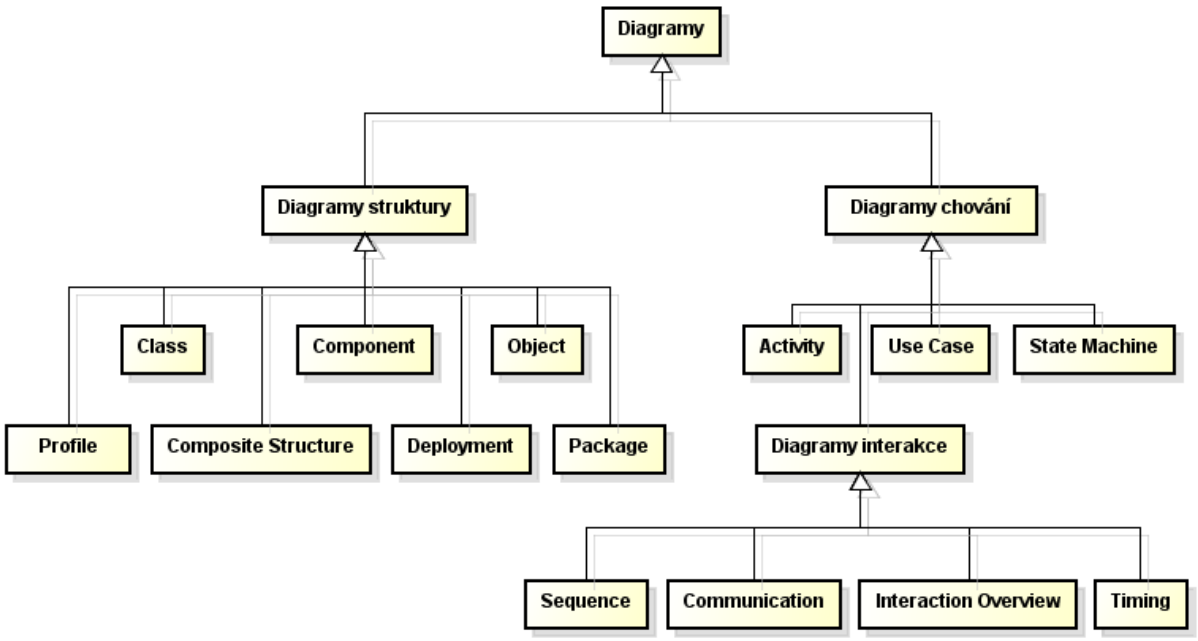
Jak bylo uvedeno výše, tak se při tvorbě infomačního systému využívají modely pro popsání funkční i datové struktury systému. Na těchto modelech lze určit komplexitu v jednotlivých fázích životního cyklu. Obsahovým dimenzím dle MMDIS náleží modely a metriky dle Tabulka 1.

Tabulka 1: UML modely a metriky dle dimenzí MMDIS

Dimenze	**Odpovídající model dle UML**
Funkce/procesy	diagram aktivit
Data/informace	diagram tříd
Organizační a legislativní aspekty	diagram organizační struktury
Pracovní, sociální a etické aspekty, aspekty lidských zdrojů	matice vlivu a zájmu (Stakeholder matrix)
Software	diagram komponent
Hardware	diagram nasazení (Deployment diagram)
Uživatelské rozhraní	modely uživatelského rozhraní (GUI model)
Bezpečnost	plán řízení rizik (Risk management plan)
Ekonomické a finanční aspekty	kalkulace nákladů dle dílčích aktivit

Zdroj: [10][16]

[2] UML (Unified Modeling Language) je soubor diagramů, využívaných pří vývoji softwaru. Standard v oblasti analýzy a návrhu informačních systémů. [6]



Obrázek 11: Druhy diagramů v UML

Zdroj: [6]

Při zkoumání komplexity je třeba se zaměřit na dimenze a fáze, které mohou složitost více či méně ovlivnit. Jsou to všechny fáze od globální podnikové strategie až po implementaci. Dimenze v jednotlivých fázích, které mají vliv na komplexitu, jsou následující:

- Funkce/procesy;
- Data/informace;
- Organizační a legislativní aspekty;
- Software;
- Hardware;
- Uživatelské rozhraní.

Dimenze obsahující pracovní, sociální a etické aspekty, aspekty lidských zdrojů, bezpečnost, ekonomické a finanční aspekty se nekvantifikují pomocí UML modelů, ale jsou posuzovány dle jejich vlivu na komplexitu, a také jak komplexita ovlivňuje je samotné. Níže bude uveden detailnější popis obsahových dimenzí a přiřazených modelů.

2.4.1. Data/informace

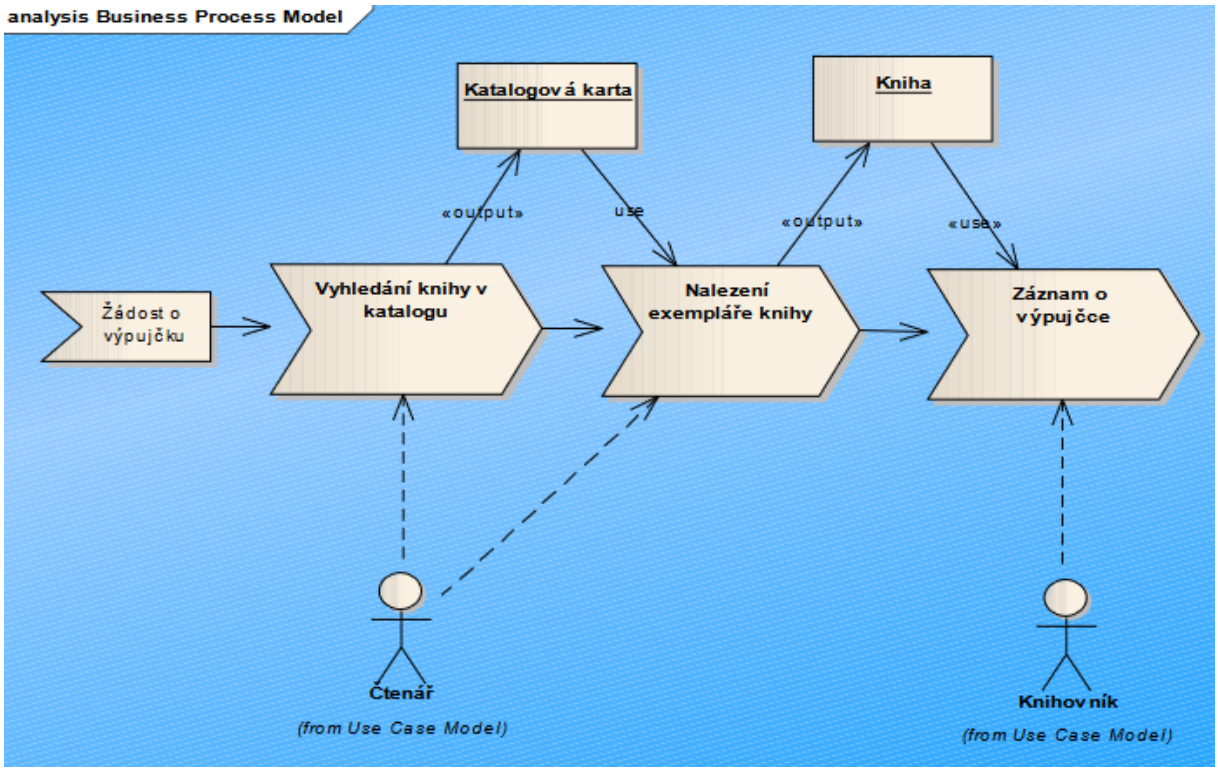
Hierarchický model organizační struktury ve fázi globální strategie podniku zahrnuje všechny oblasti organizace. Z tohoto diagramu vyplývají veškeré možné informace, které zasáhnou do informačního systému. Na základě obsažení podnikový oblastí je třeba určit velikost potřebné databáze na úrovní informační strategie. V úvodní studii je definována struktura datového úložiště, jeho velikost a počet prvků. V konceptuální úrovni je používán diagram tříd, který musí být zejména úplný, ale také přehledný. Obsahuje objekty, které program bude obsahovat se všemi atributy a metodami. Diagram tříd patří do skupiny diagramů struktur a zobrazuje statický pohled na systém. Prvky diagramu jsou třídy, které označují typy objektů systému, obsah tříd a statické vztahy mezi nimi. Z tohoto diagramu je realizována většinou vlastní relační databáze systému. Počet tabulek a relací mezi nimi určuje komplexitu v datové dimenzi dle MMDIS.

2.4.2. Funkce/procesy

V dimenzi týkající se funkcí a procesů je na úrovni konceptuálního modelu využíván diagram aktivit, který se používá pro popis dynamických aspektů systému. Znázorňuje tok řízení z jedné aktivity do další aktivity. Diagram aktivit se soustřeďuje více na proces výpočtu než na objekty účastnící se výpočtu. Znázorňuje sekvenci stavů a podmínky způsobující přechody mezi jednotlivými stavy. V diagramu pro každou aktivitu se vyskytují následující prvky:

- oddíly (partitions, swimlanes): rozdělení diagramů na více částí pro znázornění odpovědností za různé části aktivity;
- akce (action): nejprimitivnější, nejnižší prvek výpočtu, který nemůže být dále dekomponován;
- hrana, tok (edge, flow): zobrazuje přechod z jedné aktivity do další;
- objekt (object): může vstupovat do aktivit a může být jejich výsledkem;
- počáteční a finální uzel aktivity (initial, activity final): uzly pro označení počátku a konce aktivity. [1][31]

V technologické úrovni modelu lze využít Eriksson-Penker notaci, která prostřednictvím souboru stereotypů poskytuje jedinečný a výkonný způsob vizualizace komunikace obchodních procesů a potřebného toku informací v rámci organizace. Příklad jednoduchého modelu s využitím notace je na Obrázek 12.



Obrázek 12: Eriksson-Penker notace

Zdroj: [8]

Pro modelování podnikových procesů se dnes často využívá standard BPMN (Business Process Model and Notation), která obsahuje principy a pravidla pro grafické znázornění podnikových procesů a vytváří tzv. procesní diagramy. Tyto diagramy jsou založeny na flowchart „konceptu“, kdy je diagram tvořen tokem aktivit, které mají nějaké podmínky a dané pořadí. V diagramu jsou zahrnuty a zobrazeny také vnější vlivy, které nejsou součástí daných podnikových procesů, ale ovlivňují je. Tato notace je jednoduchá a intuitivní pro vlastníky procesu a vyjadřuje komplexitu procesů. [3] Tato notace bude využita v kapitole 3 pro modelování vybraného procesu.

2.4.3. Organizační členění

Třetí dimenze dle MMDIS se týká organizačního členění organizace. Tato dimenze se zabývá všemi druhy uživatelů v dané organizaci. Pro vytvoření přehledného vyjádření množiny uživatelů se využívá organizační struktura, což je hierarchické uspořádání vztahů mezi jednotlivými pracovními pozicemi v organizačním útvaru, ale obsahuje také popis vztahů s dalšími útvary v organizaci. Znázorňuje vztahy nadřízenosti a podřízenosti (pomocí tzv. Org grafů) a řeší vzájemné kompetence, vazby a odpovědnosti. Nastavuje také pravidla pro komunikaci. Slouží ke sjednocení jednotlivých podnikových činností, procesů a lidí za účelem dosažení cílů organizace. [22][23]

Z organizační struktury je následně sestaven diagram uživatelských rolí. Pro každou roli je následně nastaven druh oprávnění.

2.4.4. Software

V dimenzi zabývající se softwarem je ve fázi globální strategie rozhodnuto, jaké obchodní oblasti bude informační systém zahrnovat. Ve fázi informační strategie je rozhodnuto o druhu informačního systému nebo modulu. V rámci úvodní studie je pomocí diagramu komponent popsána struktura systému. Diagram komponent obsahuje logické a fyzické komponenty použité v systému.

Využívají se dva typy komponent:

- **základní komponenta**: modulární část systému, která zapouzdřuje svůj obsah a její projev je nahraditelný;
- **obalová komponenta**: rozšiřuje základní komponentu o možnosti seskupování, komponenta může dědit a importovat své členy.

Dalším prvkem diagramu je tzv. **poskytovaný interface**, který je implementován přímo danou komponentou a **požadovaný interface** který je typem požadovaného portu, nebo je určený závislostí přímo z dané komponenty. **Port** umožňuje organizování interfejsů do skupin. **Delegační konektor** propojuje externí rozhraní komponenty s vnitřní částí komponenty. **Montážní konektor** spojuje požadovaný a poskytovaný interfejs. [5]

Na technologické úrovni vyjadřuje komplexitu počet zpracovatelných transakcí a nabízených funkcí. Pro zhodnocení software z pohledu jeho složitosti je ve fázi implementace využita metoda SLOC (Source lines of code). Jedná se softwarovou metriku pro měření velikosti počítačového programu založená na počtu řádků zdrojového kódu. [26]

2.4.5. Hardware

V dimenzi týkající se hardwaru je navržena velikost a struktura datového centra, a také topologie serveru. Zde je mírou komplexity počet prvků datové a síťové struktury. Ve fázi globální analýzy a návrhu je definována tzv. **konfigurační položka.** Konfigurační položka je jakákoli komponenta služby, prvek infrastruktury nebo jiná položka, kterou je potřeba spravovat, aby bylo zajištěno úspěšné doručení služby. Konfigurační položky se liší ve složitosti, velikosti a typu. Mohou existovat v rozsahu od celých služeb, které jsou tvořeny hardwarem, softwarem a dokumentací, až po jediný programový modul nebo vedlejší hardwarovou komponentu. Nejnižší úrovní konfigurační položky je obvykle nejmenší jednotka, která se mění nezávisle na jiných komponentách. [14]

Pro správu konfiguračních položek slouží CMDB (Configuration management database). CMDB je zdrojem informací pro podporu řízení ICT, jako jediný obsahuje komplexní přehled o veškerých existujících komponentách ICT. CMDB čerpá informace o konfiguračních položkách primárně pomocí automatických funkcí, kdy nástroj získává informace za použití modifikovatelných skriptů (typicky údaje o serverech, telekomunikačních zařízeních, databázích atd.). [15]

V dimenzi týkající se hardwaru se jako konfigurační položkou zabýváme veškerým hardwarem využitým v rámci daného informačního systému. Komplexita je hodnocena pomocí počtu prvků a vazeb mezi nimi v diagramu nasazení, kde jsou zakresleny jednotlivé konfigurační položky.

2.4.6. Uživatelské rozhraní

V dimenzi vztahující se k uživatelskému rozhraní jsou pomocí digramů případů užití a scénářů definováni jednotliví uživatelé a jejich případy užití systému. Případ užití zobrazuje chování systému nebo jeho části z hlediska uživatele. Definuje funkcionalitu, kterou by měl navrhovaný systém umět. Každý případ užití představuje použití systému účastníkem, každý případ užití má jméno. Účastník reprezentuje to, co se systémem komunikuje a interaguje. Může to být osoba, hardware nebo jiný systém, který přijímá nebo předává systému informace. V diagramu je také znázorněna hranice systému. Dále obsahuje vztahy mezi aktéry a užitími i mezi užitími mezi sebou například v případě opakované činnosti při různých užitích. Pro jednotlivé případy užití je třeba specifikovat podrobnosti pomocí scénářů, což je textový popis, který krok po kroku popisuje interakce mezi aktérem a systémem. [30]

Dále jsou navrženy počty jednotlivých obrazovek systému a je sestaven GUI model. Uživatelské rozhraní má zásadní dopad na práci uživatelů systému. Složité a nepřehledné obrazovky a ovládání může vézt až k averzi zaměstnanců systém využívat, proto je nutné při vývoji grafického rozhraní úzce spolupracovat s uživateli a nalézt vhodné řešení, které uspokojí v co největší míře uživatelské požadavky a zároveň nezpůsobí vysokou složitost systému. Složitost této dimenze ve fázi implementace je posuzována na základě počtu obrazovek a položek na obrazovce.

2.4.7. Bezpečnostní ekonomické a finanční aspekty

V této dimenzi se bezpečností zabývá plán řízení rizik, kde jsou odhalovány rizika a je stanovena strategie jejich řešení, ve vztahu ke komplexitě. Je zde možno kvantifikovat počet možných rizik a jejich závažnost. Kvantifikace finanční náročnosti spočívá v souhrnu všech nákladů na informační systém v jednotlivých fázích životního cyklu.

Přiřazení modelů a metrik jednotlivým obsahovým dimenzím dle MMNDIS v rámci životního cyklu je v Tabulka 2.

Tabulka 2: Modely a metriky dle obsahových dimenzí MMNDIS

	Data/ informace	Funkce/ procesy	Organizační členění	Software	Hardware	Uživatelské rozhraní	Komplexita
Konceptuální úroveň	Diagram tříd	Diagram aktivit	Organizační struktura	Diagram komponent	Diagram nasazení	Use case diagram	∑
Technologická úroveň	Relační databáze	Eriksson-Penker notace	Organizační struktura	Transakce a funkce	Topologie sítě	GUI modely	∑
Implementační úroveň	Databáze	BPMN	Uživatelské role	SLOC	Počet konfiguračních položek	Počet obrazovek	∑
Provoz a údržba	Databáze	BPMN	Uživatelská oprávnění	SLOC	Počet konfiguračních položek	Počet položek obrazovek	∑

Zdroj: vlastní zpracování

2.5. Kvantifikace komplexity

V předchozí části byly určeny UML modely pro jednotlivé dimenze v jednotlivých fázích životního cyklu informačního systému dle MMDIS. Složitost systému **C(s)** je dána součtem složitostí všech obsahových dimenzí **C(Di).** Na základě tohoto je možno kvantifikovat komplexitu v jednotlivých fázích životního cyklu. Nejprve definujme složitost **C** systému **S** ve fázi **f** jako počet entit (prvků a vazeb) UML modelů jednotlivých obsahových dimenzí **D.** Písmenem **d** označme celkový počet uvažovaných dimenzí. Složitost **CfDi** je pak počet entit dimenze **i** ve fázi **f.** Výsledný vzorec (1) je:

$$C_f(S) = \sum_{i=1}^{d} C_f \mathrm{D_i} \qquad (1)\ [10]$$

Vzorec (1) je použitelný pro kvantifikaci složitosti systému. Je tedy možno srovnat různé informační systémy nebo alternativy řešení dané situace. V systému lze rozlišit dva druhy komplexity, a to tzv. zbytečnou (nadbytečnou) a nezbytnou. Zbytečná zahrnuje prvky a vazby, které nejsou třeba pro zajištění daných vlastností. Nezbytná neboli žádoucí je minimální množství prvků a vazeb pro dosažení určitých vlastností sytému.

Pomocí vzorce (1) zjistíme složitost daného systému v daném čase. V rámci života informačního systému dochází k jeho úpravám, jako je: přidání řádku do tabulky, vytvoření nové databázové tabulky nebo rozšíření uživatelského rozhraní o nový prvek. Je zřejmé, že s nárůstem prvků roste také komplexita. S postupem času dochází k postupnému růstu, který může v krajním případě způsobit nefunkčnost systému. Na nekontrolovatelný růst komplexity má zejména vliv neustále přidávání prvků a zároveň neodstraňování již nepoužívaných prvků a vazeb. V případě, že není informační systém cíleně zjednodušován, tak během jeho životního cyklu roste jeho komplexita. V praxi je mazání prvků systému velmi nebezpečné, neboť hrozí ztráta informace, která je uchována například v nějaké vazbě na další prvek.

Přidání jednoho prvku nezvýší komplexitu pouze o 1. S daným prvkem vznikne minimálně jedna vazba. Vzorec (2) popisuje zvýšení komplexity při přidání jednoho prvku.

$$\Delta C(S) = \sum_{i=1}^{d-1} K_{ji} \qquad (2)\ [10]$$

V rovnici (2) je **d** počet uvažovaných dimenzí a **Kji** je počet entit dimenze **i**, které je třeba přidat, aby byla zachována logická a funkční konzistence modelů, po přidání jednoho prvku v dimenzi **j**.

Příkladně po přidání jednoho prvku v dimenzi uživatelského rozhraní (pole hodnot) vznikne v dimenzi procesů vice jak 5 nových entit: pro zadání a údržbu hodnot, zobrazení, změnu a vyhodnocení tohoto pole. Obdobně se přidání tohoto prvku projeví i v dalších dimenzích.

Při přidání více prvků do více dimenzí dojde k multiplikacím, které komplexitu zásadně zvýší. Rovnice (3) popisuje, jak se změní celková komplexita přidáním **n** prvků do systému, tedy $\mathbf{n_j}$ prvků do každé dimenze j navýší tedy celkovou složitost následovně:

$$\Delta C(S) = \sum_{j=1}^{d} \sum_{i=1}^{d-1} K_{ji} * n_j \text{ , platí } n = \sum_{j=1}^{d} n_j \qquad (3)\ [10]$$

Z výše uvedeného je patné, že každé rozšíření globální podnikové strategie o další prvek implikuje nárůst počtu podnikových procesů, informatických služeb a dalších prvků, které v dalších fází životního cyklu způsobí nárůst komplexity IS.

2.6. Zhodnocení komplexity v jednotlivých fázích abstrakce vývoje IS

Při vývoji informačního systému je třeba brát ohled na jeho budoucí komplexitu, neboť ta má zásadní vliv na fungování systému a ovlivňuje také náklady na systém. Komplexita ovlivňuje náklady jak na vývoj IS, tak i na jeho následný provoz a případný reengineering. Z hlediska uživatele by měl být systém tak složitý, aby obsáhl veškeré potřebné procesy, ale zároveň byla jeho struktura uživatelského rozhraní dobře čitelná a pro uživatele přívětivá.

2.6.1. Konceptuální úroveň

V kapitole 1.4. bylo uvedeno jaké fáze abstrakce se používají při vývoji informačního systému. Prvním krokem je tedy vytvoření modelu reality (konceptuální úroveň), tedy vytvoření modelů procesů, které mají být do systému zavedeny. Před automatizováním procesů pomocí informačního systému je třeba se nejdříve zaměřit na vlastní procesy a posoudit jejich komplexitu ještě před jejich zavedením do systému. Pro posouzení komplexity procesů lze přistupovat několika způsoby.

Lze posuzovat proces dle obtížnosti jeho popisu. Jako mírami pro posouzení může být použita tzv. minimální délka popisu, entropie, algoritmická komplexita nebo fraktální dimenze. O složitosti procesu vypovídá také obtížnost jeho vytvoření. V rámci tohoto se používají metody pracující s časem, náklady a energií, nutnou pro vytvoření procesu jako je časová výpočetní komplexita, termodynamická a logická hloubka a množství nákladů. Na komplexitu procesu má vliv také stupeň jeho organizační struktury. Obtížnost popisu organizační struktury a množství informace rozdělená mezi jednotlivé části určuje stupeň organizovanosti procesu.

Mezi metody pro určení složitosti organizační struktury procesu patří hierarchická komplexita, délka schématu, sofistikovanost nebo algoritmus vzájemné výměny informací.

Vysoká komplexita procesů ovlivní jejich model a složitost se následně promítne do dalších fází vývoje systému. Riziky vysoké komplexity v této fázi je budoucí složitá práce se systémem, velké nároky na uživatele, hardware i databázi. Dalším rizikem, které se projeví až v případě nutnosti systém modifikovat vychází z faktu, že s rostoucí komplexitou klesá modifikovatelnost systému, a tím rostou náklady na provádění změn, při kterých navíc hrozí ztráta dat z důvodu úprav ve složité datové struktuře.

Hlavním rizikem nízké komplexity v této fázi je, nepostihnutí všech žádaných procesů nebo jejich variant. V případě, že není nedostatek v podobě chybějícího procesu nebo funkce včas odhalen a systém je zaveden, budou uživatelé nuceni buď využívat další SW nástroj, který funkcionalitu obsahuje, nebo bude nutné tuto funkci do systému zavést dodatečně. Obě varianty s sebou nesou další dodatečné nároky na čas a finance.

Výstup z konceptuální úrovně by, měl obsahovat pouze prvky a vazby nezbytné pro zajištění všech žádaných funkcí na podporu procesů definovaných v rámci informační strategie. V modelu by měla být obsažena pouze **nezbytná komplexita.**

2.6.2. Technologická úroveň

V další fázi návrhu je z konceptuálního modelu vytvořen model technologický. V rámci tohoto jsou vytvořeny jednotlivé funkcionality, jsou vymezeny uživatelské role, je definován hardware, datová struktura a grafické uživatelské rozhraní. Technologický model je tvořen již na vybrané softwarové řešení. Organizace v tuto chvíli může volit mezi realizací systému pomocí masivních řešení jako je například SAP a využít jeho předdefinovaných funkcionalit, a provézt pouze úpravy, definovaných vlastností a ihned přikročit k zavádění systému. Další možností je vytvoření systému na míru a jeho naprogramování systému přesně dle technologického modelu.

Z hlediska nadbytečné komplexity je varianta vývoje systému na míru lepším řešením než použití masivního řešení, a to zejména z pohledu uživatele. Při tvorbě systému přesně dle výstupů z konceptuální úrovně se nebude komplexita v technologické úrovni zvyšovat.

Při snaze „napasovat“ hotový systém na procesy v podniku musí nezbytně k růstu **nadbytečné komplexity** dojít, neboť v programu budou funkce, které organizace nepotřebuje a budou uživateli „překážet“ při práci. Další možný negativní případ nastane, když bude nějaká

funkcionalita scházet a bude nutno upravit podnikové procesy dle systému, což s sebou nese rizika neefektivního využití pracovní doby zaměstnanci nebo vzniku chyb.

V této fázi se vysoká komplexita systému projeví zejména složitostí uživatelského rozhraní a dobou zpracování procesu systémem a vysokými hardwarovými nároky. Zásadní dopad na organizaci má nadbytečná složitost uživatelského rozhraní, neboť má přímý vliv na práci zaměstnanců. Musí-li pracovník provést příliš mnoho operací (například kliknutí nebo přihlášení), stává se jeho práce neefektivní a rostou náklady na jeho práci, neboť stráví vykonáním daného úkonu více času, než je nutné. Vysoká složitost uživatelského rozhraní prodlužuje dobu zácviku nových zaměstnanců a může vézt k odporu zaměstnanců systém používat. Také v případě, že proces vyžaduje vysoké nároky na hardware (velký počet operací, které musí výpočetní technika vykonat) má negativní vliv na efektivitu práce (dlouhé odezvy systému) a zvyšuje nestabilitu systému (hrozí přetížení výpočetní techniky), proto musí organizace adekvátně dimenzovat použitý hardware, což opět zvyšuje náklady.

Nízkou nadbytečnou komplexitu bude dosahovat systém naprogramovaný na míru dané organizace. Tyto systémy ovšem s sebou nesou riziko nízké modifikovatelnosti při potřebě změn v budoucnosti. Například při rozšíření informační strategie bude velmi těžké a nákladné systém přizpůsobit a v extrémním případě bude nutno vyvinout systém nový. Masivní řešení jako je SAP je z tohoto pohledu variabilnější.

2.6.3. Implementační úroveň

Implementační úroveň v sobě zahrnuje vlastní programování a následné testování softwaru. Dále je na této úrovni provedeno zavádění systému do praxe. Možnosti zavádění systému byly uvedeny v kapitole 1.3.4. Při výběru postupu zavádění systému je třeba brát ohled na jeho uživatelskou složitost. Nárazová strategie je vhodná pouze pro systémy s nízkou mírou složitosti uživatelského rozhraní, kdy se uživatel v systému rychle zorientuje a riziko chyb je minimální. Postupné a pilotní zavádění je vhodné u komplexnějších systémů, kdy je složitost větší, a proto je uživateli „dávkována“ postupně a ten tak má čas se na systém adaptovat. Pro velké systémy s vysokou komplexitou je ze zmíněných nejvhodnější strategie souběžného provozu. Je však bezpodmínečně nutné mít zacvičování uživatelů pod kontrolou, neboť hrozí, v důsledku existence starého systému odkládání učení se nového systému, což prodlužuje souběžný provoz, který je nákladný.

Vysoká složitost systému má také vliv na dobu programování a zejména testování. V systému, kde existuje mnoho prvků a vazeb s vnitřními závislostmi je na otestování všech variant třeba dostatek času pro snížení pravděpodobnosti chyby v reálném provozu. To s sebou

nese opět vysoké náklady. Nízká komplexita systému nese riziko podcenění doby testování i doby potřebné pro zaučení pracovníků. To může způsobit chyby v provozu relativně jednoduchého systému.

2.6.4. Provoz a údržba

Po zavedení systému následuje časově nejdelší etapa života informačního systému, a to jeho rutinní provoz, který je spojený s pravidelnou údržbou. Během provozu je často ještě systém upravován dle zjištěných nedostatků, případně rozšířen z tohoto důvodu komplexita většiny systémů sama roste. K jejímu cílenému snižování například v důsledku odstranění nepoužívaných procesů však zpravidla nedochází nebo to není možné například z důvodu existence záznamů v závislých tabulkách, zároveň je tato činnost velmi riziková a nákladná. V této části životního cyklu má vysoká komplexita přímý vliv na náklady spojené s údržbou. V případě změny informační strategie může být přikročeno k reengineeringu systému, ten však u systému s vysokou komplexitou není snadný ani levný a je třeba zvážit, zda není výhodnější vyvinout systém nový.

V Tabulka 3 jsou uvedeny nositelé komplexity v jednotlivých obsahových dimenzích ve všech fázích životního cyklu IS.

Tabulka 3: Nositelé komplexity a jejich vliv v rámci životního cyklu IS

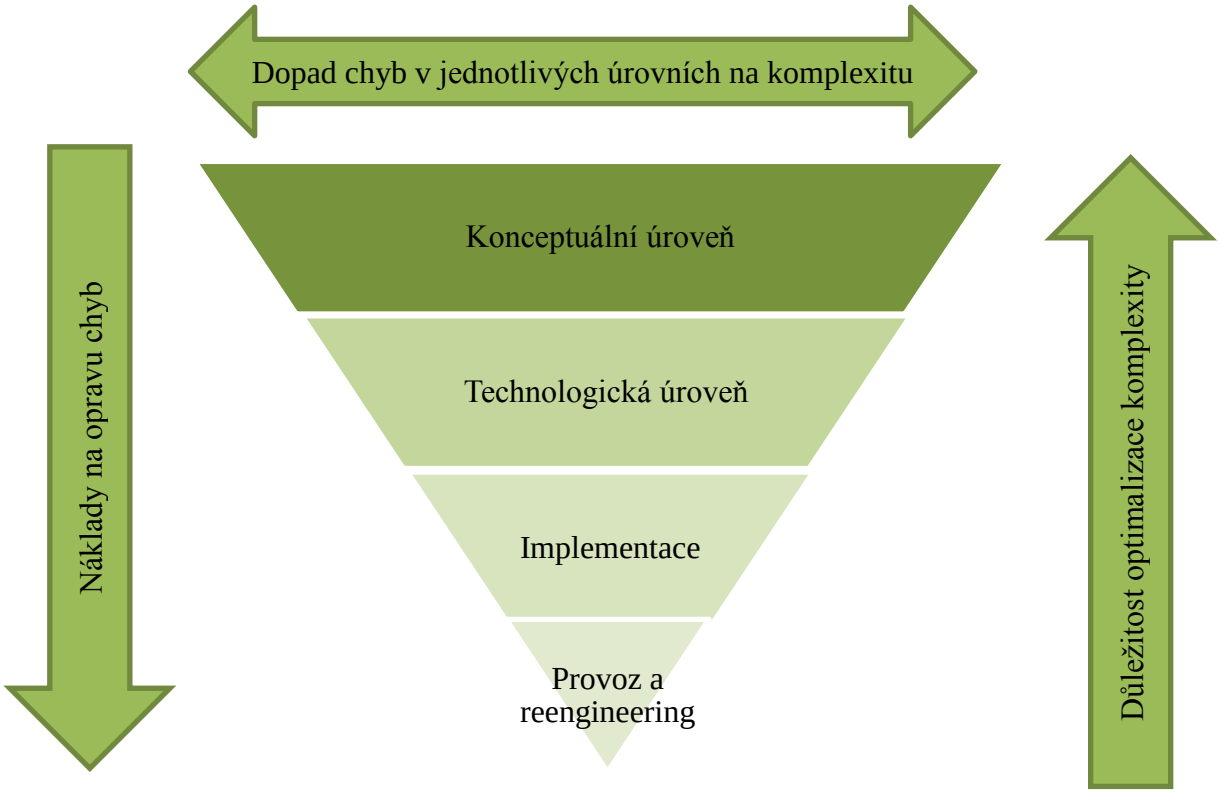
Úroveň/fáze abstrakce	Nositel komplexity	Vliv na komplexitu
Konceptuální úroveň	**Organizační struktura podniku**	Velikost a členitost organizační struktury řešené informačním systémem má zásadní vliv na jeho komplexitu. Organizační struktura má vliv na počet modulů IS, velikost databáze, počet procesů, počet uživatelů a počet HW komponent. Struktura IS odpovídá struktuře podniku, proto je třeba podnikovou strukturu optimalizovat ještě před vývojem IS.
	Procesy	Procesy zaváděné do IS jsou jeho funkční a datovou náplní. Složité neefektivní procesy není vhodné zavádět do IS. Každý proces je nutno posoudit z hlediska komplexity před jeho zavedením.
	Uživatelé	Počet uživatelů systému vyplývá z procesů a organizační struktury, však ne všichni musí být v systému zavedeni. Systém by měl automatizovat procesy prováděné lidmi. Počet uživatelů musí být optimální pro úspěšné provádění procesů s podporou IS, příliš mnoho uživatelů zvyšuje komplexitu.
	Software	V této fázi je třeba rozhodnout, zda bude systém realizován pomocí dostupného hotového SW řešení nebo bude IS vyvinut na míru podnikových procesů.
	Hardware	Volba HW řešení ovlivňuje komplexitu z pohledu výpočetní kapacity. HW je nutno volit s ohledem na předpokládanou dobu provozu IS. Předpokladem je zvyšování nároků na HW v čase. Nevýkonný HW v budoucnu zpomalí práci s IS. Velikost a hustota sítě hardwarových komponent ovlivňuje stabilitu a rychlost IS.
Technologická úroveň	**Datová struktura**	Komplexitu ovlivňuje zvolený typ databáze, počet tabulek a jejich atributů, relací a nastavená integritní omezení.
	Funkční struktura	Sestavení optimálního diagramu aktivit z daného procesu zásadně ovlivní komplexitu systému. Příliš složitě namodelovaní procesu bude mít za následek velké množství funkcí systému, které se dále promítnou do složitého uživatelského rozhraní a do velikosti vlastního kódu vlastního SW. Je třeba postihnout celý proces v co možná nejjednodušší formě.
	Komunikační struktura	Vymezení toho jak a s kým nebo čím bude systém komunikovat je důležité pro všechny účastníky jednotlivých procesů. Systém muže komunikovat buď s uživateli, nebo s dalšími SW v organizaci. Vliv na komplexitu má zejména počet předaných informací.

		Informace, které nikdo nepoužívá, zvyšují nadbytečnou komplexitu.
	Zvolené SW řešení (programování „na míru“ / nákup hotového řešení)	Při nákupu hotového systému se provádí jeho „napasování“ na procesy v podniku. Každý již hotový SW má omezené možnosti modifikace (některé prvky nelze odebrat jiné zase přidat), toto opět negativně ovlivňuje komplexitu a práci uživatelů se systémem. IS vytvářený přesně dle organizace, obsahuje jen takovou komplexitu, která je obsažena ve vyšší úrovni abstrakce.
	HW struktura	Dle velikosti datové struktury a počtu uživatelů, je třeba zvolit a navrhnout komplexní hardwarovou síť. Komplexitu zvoleného hardware ovlivňuje jeho struktura (počet serveru, datové pole, klientské PC, huby nebo switche)
	Grafické rozhraní	Grafické rozhraní má zásadní vliv na komplexitu z hlediska uživatele, komplexita GUI modelů ovlivní každodenní práci uživatelů. Počet obrazovek a položek na obrazovce vyplývají z datové a funkční struktury. Grafické rozhraní musí být uživatelsky přívětivé a nesmí obsahovat nadbytečné prvky, které stěžují uživateli práci. Složitost rozhraní je volena dle hierarchie řízení v podniku (manažerské IS mají jednodušší rozhraní než IS používaný ve „výrobě“).
	Programovací jazyk	Zvolený programovací jazyk, zvolený framework (API). Každý programovací jazyk má určité možnosti, jak realizovat danou proceduru. Jazyk, kde je žádaná procedura (funkce) již nadefinovaná je „jednodušší“ a než ten kde je třeba žádanou proceduru programovat ručně pomocí několika řádků příkazů.
Implementace	**Struktura kódu**	Jak je strukturován kód, je důležité zejména v případě, kdy je třeba v kódu něco měnit. V případě, že je program napsán nepřehledně, nečitelně nebo je těžko pochopitelný (není dokumentace, chyby komentáře) je tento program obtížně modifikovatelný. Zde komplexitu ovlivní vlastní programátor SW.

	Implementace hotového IS	Hotový IS si přináší svoji komplexitu od svého výrobce. Cílem implementace je tuto komplexitu optimalizovat na potřebnou velikost (počet entit a atributů, optimální počet prvků UI, odstranění nepotřebných funkcí). Snížení komplexity je často možné pouze skrytím položek na obrazovkách uživatelského rozhraní. K optimalizaci databáze nebo zdrojového kódu většinou nedochází.
Provoz, údržba, reengineering		V provozní fázi je zanesena komplexita ze všech fází předchozích a veškeré chyby se zde projeví. Při optimálním provedení všech vývojových kroků je i komplexita optimální a v čase nedochází k jejímu zvyšování. Zanedbání optimalizace v některém kroku bude mít negativní následek při provádění, byť jen drobných změn systému v rámci jeho provozu.

Zdroj: vlastní zpracování

Tabulka 3 shrnuje většinu aspektů ovlivňujících komplexitu IS v jeho životním cyklu. V každé fázi je třeba mít na paměti, že komplexita na nižší úrovni je exponenciálně důležitější než komplexita na úrovni vyšší. Špatně zvolená architektura na konceptuální úrovni se například při rozšíření oblasti podnikání organizace projevuje menší elasticitou, což má vliv na náklady a negativně ovlivňuje komplexitu i přes to, že další kroky v následujících úrovních byly optimální. Při dobře zvolené architektuře, ale například špatné definici prvků systému, lze provézt nutné změny s menšími náklady bez růstu komplexity. Celkově lze shrnout, že chyby provedené v nižších úrovních mají menší dopad na komplexitu i náklady organizace. Graficky toto shrnuje Obrázek 13.



Obrázek 13: Komplexita v jednotlivých fázích abstrakce

Zdroj: vlastní zpracování

3. ANALÝZA INFORMAČNÍHO SYSTÉMU SAP V SEV.EN EC A.S.

Tato kapitola se bude dále věnovat komplexitě ERP systému SAP/R3 ve společnosti Sev.en EC a.s. Nejprve se seznámíme s předmětem podnikání společnosti a následně se bude práce věnovat systému SAP, který je v organizaci implementován. Detailně se vzhledem k velkému rozsahu systému budeme věnovat modulu pro údržbu a opravy (PM), kde bude naznačena multiplikace komplexity při přidání jednoho prvku do organizační struktury a dále bude zhodnocena komplexita uživatelského rozhraní.

3.1. Popis podniku

Společnost Sev.en EC a.s. se zabývá výrobou elektřiny v klasické elektrárně Chvaletice. Provoz se nachází v Polabí nedaleko Pardubic. Byl postaven v letech 1973–1979 na území bývalých Mangan-kyzových závodů, v nichž právě tehdy končila těžba pyritu. S výstavbou elektrárny souviselo dobudování Labské vodní cesty, protože severočeské hnědé uhlí, které se ve Chvaleticích spaluje, sem bylo do poloviny roku 1996 dopravováno z Lovosic po vodě. Celkový instalovaný výkon elektrárny je 820 MW a tvoří ho čtyři 205MW bloky. Výkon je vyveden dvěma 400 kV linkami do rozvodny Týnec nad Labem.

Podnik kromě elektřiny také obchoduje s plně certifikovanými vedlejšími energetickými produkty spalování uhlí (struska, popílek, energo sádrovec, stabilizát) a dodává teplo do přilehlé průmyslové zóny a města Chvaletice.

3.2. Analýza IS

V podniku je od roku 2013 implementován systém SAP. Jsou použity následující moduly:

AM – evidence majetku;

FI – finance a účetnictví;

MM – logistika: nákup a sklady;

SD – logistika: prodej a expedice;

PM – údržba;

CO – controlling;

HR – personalistika.

Oblasti jednotlivých modulů vyplývají z jejich názvu. Blíže se analýza bude věnovat pouze modulu pro údržbu (PM).

3.2.1. Modul PM

V rámci modulu PM lze všeobecně rozlišovat dva druhy údržby**. Preventivní údržbu**, kde jsou tvořeny pečlivé plány, například rok dopředu, a je generován plán údržby. **Korektivní / všeobecná** údržba, vychází z toho, že došlo k závadě na nějakém zařízení a je nutné je opravit, co nejrychleji, aby nedošlo ke zdržení výroby, což je v energetice velmi klíčový faktor, neboť každá hodina výpadku výroby bloku o výkonu 205MW s sebou nese významnou finanční ztrátu, a to zejména z důvodu toho, že vyráběná elektřina je již dopředu zobchodovaná v rámci trhu s elektrickou energií.

Modul PM používají zaměstnanci **provozu a péče o zařízení (PoZ).** Je využíván pro údržbu veškerého zařízení v elektrárně. Zahrnuje jak údržbu zařízení související s výrobou, tak i ostatních vybavení jako jsou budovy a komunikace. V rámci oddělení péče o zařízení jsou v systému zavedeny tyto plánovací skupiny: stavební, elektro, kotelna, měření a regulace, odsíření, strojovna, vodní hospodářství a palivové hospodářství. Opravy v podniku probíhají pomocí zakázek. V těch je uvedeno, co je třeba opravit a kdo opravu provede. Údržba je prováděna pomocí externích firem a dodavatelů. Správa zakázek probíhá v PM modulu systému SAP.

Popis procesu opravy závady

Zaměstnanci provozu při zjištění závady na zařízení vystaví hlášení, ve kterém specifikují závadu. Dále vypíší a umístí na místo závady poruchovou visačku. Zaměstnanec PoZ (dle plánovací skupiny) vystaví z hlášení zakázku na údržbářskou firmu. Ta převezme papírovou formu zakázky a jde za pracovníkem provozu, který potvrdí, zda lze na zařízení pracovat (podpis) a v systému změní stav zakázky na „V realizaci“. Po ukončení opravy jde údržbář k pracovníkovi provozu, který provede kontrolu opravy a v případě, že je oprava provedena řádně, provede stvrzení dokončení opravy podpisem zakázky a v systému SAP změní stav zakázky na „Z realizace“. Pracovník údržby následně donese zakázku danému pracovníkovy PoZ a ten provede ukončení zakázky. Ukončená zakázka je dále zpracovávána v modulu CO. Každá zakázka je vztažena na daný technický objekt. Proces opravy je na diagramu v příloze A.

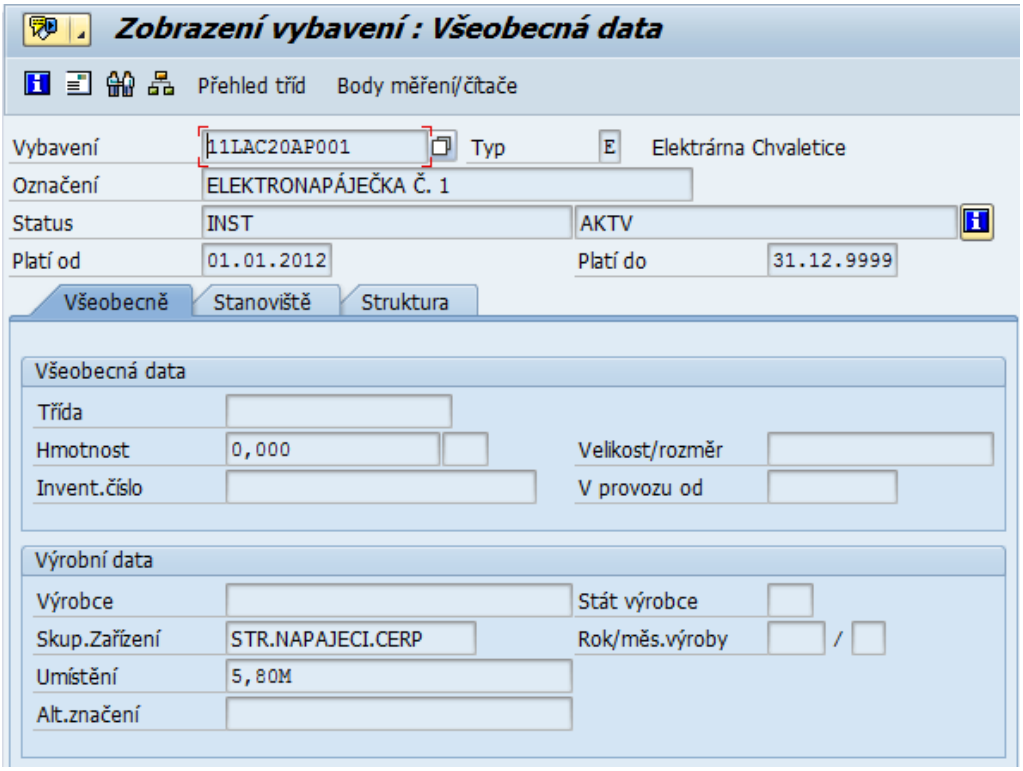
Pro evidenci technických objektů údržby jsou v systému SAP použita označení „Technické místo“ a „Vybavení“. Každá závada je vztažena k danému „Vybavení“. To představuje konkrétní detailní část zařízení. Každé vybavení má svůj specifický kód KKS. Při tvorbě hlášení je třeba zadat KKS poškozeného zařízení, dle tohoto je v systému automaticky hlášení zařazeno

do dané plánovací skupiny danému pracovníkovi PoZ, dále dojde k automatickému vyplnění dalších polí formuláře na Obrázek 15 níže.

Technické místo je objekt v technické struktuře a představuje úsek, na který je možno instalovat vybavení. Každé technické místo je definováno a spravováno ve vlastním kmenovém záznamu a vytváří vlastní historii údržby. Systém SAP umožňuje znázornění různých úrovní hierarchie technických míst a vybavení v tzv. stromové struktuře.

Příkladem technických míst mohou být větší celky, které zůstávají v celé době životnosti stabilní (např. stanice, potrubí, vedení, transformovny, budovy atd.).

Vybavení představuje individuální fyzické zařízení, které je z hlediska údržby samostatně udržováno a má vlastní kmenový záznam. Vybavení mohou být v průběhu životnosti instalována na různých technických místech. Kromě toho je možné, že po určitou dobu jsou nainstalována na fiktivních technických místech („skladech“), když se nacházejí ve skladu nebo v opravě. Jednotlivé použití vybavení během jeho životnosti je dokumentováno systémem a každé vybavení má svou vlastní historii. Tabulka údajů pro jednotlivá vybavení je na Obrázek 14 níže.

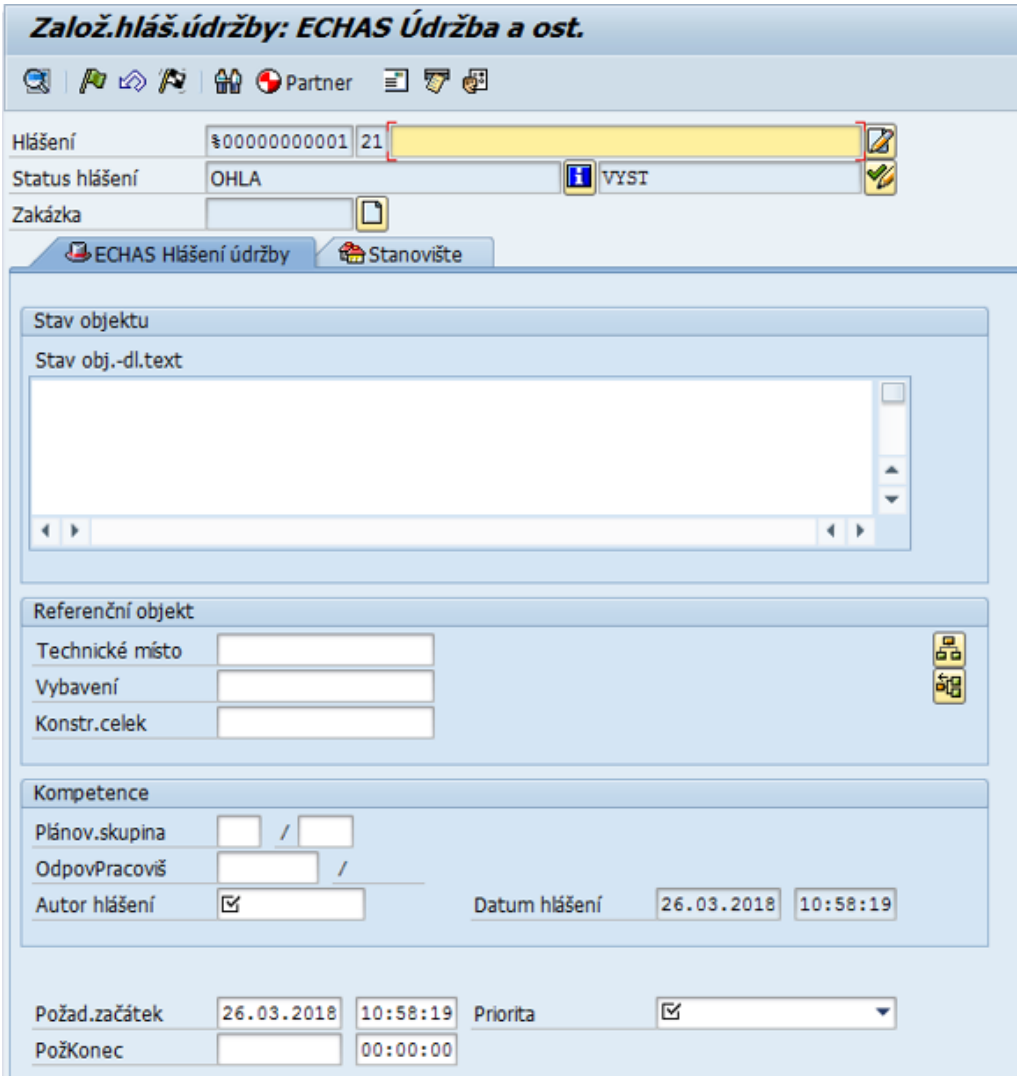


Obrázek 14: Příklad vybavení v SAP

Zdroj: SAP

3.2.2. PM Hlášení a PM Zakázky

Hlášení údržby, jak bylo zmíněno, slouží zaměstnancům provozu pro ohlášení závady na zařízení, pomocí kterého je upozorňováno oddělení údržby na potřebné opatření a pomocí, kterých se dokumentuje vykonaná práce. Každé hlášení údržby má hlavičku, která obsahuje nejdůležitější informace sloužící k identifikaci a správě. Lze v něm popsat požadované výkony údržby, důvody hlášení a nutné kroky pro zpracování hlášení. V hlášení je třeba zadat prioritu, která určuje, jak je oprava závažná. Obrazovka pro tvorbu hlášení je na Obrázek 15.

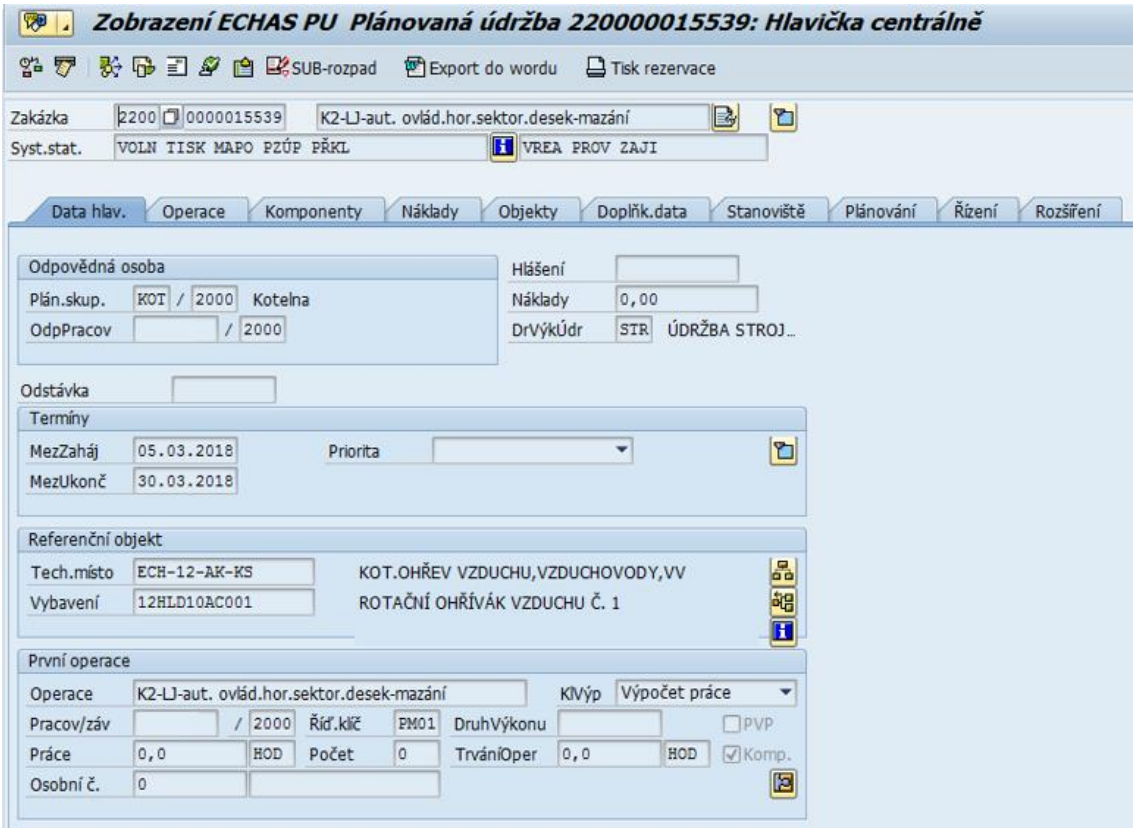


Obrázek 15: Tvorba hlášení

Zdroj: SAP

Zakázky slouží k správě a řízení údržbové činnosti. Obrazovka je na Obrázek 16 níže. U každé zakázky se určuje druh údržby, na kterou je vytvořena. Rozlišuje se nahodilá údržba (poruchy rychle k odstranění), plánovaná údržba (preventivní opravy, revize), ostatní (zakázka vytvořena z jiných důvodů než opravy například převoz materiálu…) nebo tzv. pevná cena

(zakázky vystavené v rámci generálních oprav velkých technologických celků, kdy je stanovena pevná cena za celé dílo). PM Zakázka se může vytvořit přímo z PM Hlášení nebo samostatně, bez vazby na PM Hlášení. K jednomu PM Hlášení dovolí SAP založit pouze jednu PM Zakázku.



Obrázek 16: Tvorba zakázky

Zdroj: SAP

3.3. Analýza komplexity

Na zjištěná fakta budou nyní aplikovány přístupy z kapitoly 2.4. Data jsou čerpána z implementačního dokumentu při zavádění systému v roce 2013 a od systémového analytika společnosti. Data týkající se uživatelského rozhraní byla získána rozhovory s uživateli modulu. V rámci analýzy uživatelského rozhraní bude odhalena nadbytečná komplexita v podobě prázdných a redundantních polích formulářů v rámci procesu tvorby zakázky na údržbu. Další část analýzy bude zaměřena na multiplikační efekt komplexity. Cílem bude nalézt multiplikátor komplexity při provádění změn v jednotlivých úrovních abstrakce.

3.3.1. Komplexita uživatelského rozhraní

Pro proces tvorby zakázky na údržbu zařízení elektrárny jsou využity uživateli 3 formuláře: Vybavení, Hlášení údržby, Zakázka. Pro zhodnocení komplexity byly sumarizovány veškeré položky těchto formulářů. Dále byly jednotlivé položky rozděleny povinné, automaticky vyplňované a nepoužívané. Povinná pole jsou taková, která jsou nutná pro úspěšný průběh procesu. Automatická pole, jsou taková, která systém vyplní po zadání daného povinného pole. Poli nepoužívanými se rozumí taková, která zůstávají nevyplněná a nejsou pro proces nezbytná. Duplicitní pole jsou taková, která obsahují informaci, která je již uvedena v jiném poli stejného formuláře například na jiné záložce. Výsledek tohoto rozdělení je v Tabulka 4 níže.

Tabulka 4: Počty položek uživatelského rozhraní

	Vybavení	Hlášení	Zakázka
celkem	34	35	80
povinných	10	5	26
automatických	5	23	32
nepoužívaných	16	6	15
duplicitní	3	1	7

Zdroj: vlastní zpracování

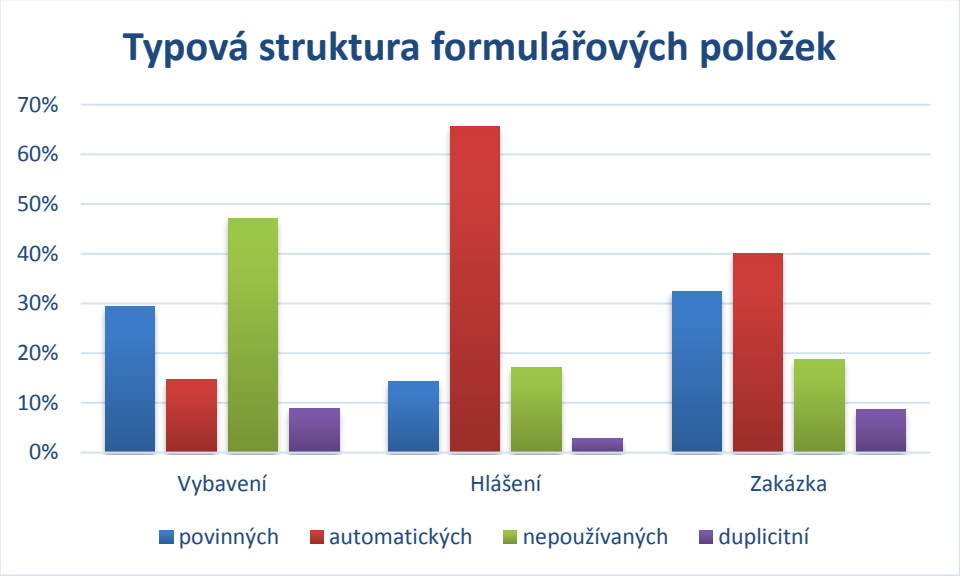
Procentuální vyjádření počtu jednotlivých typů položek je v Tabulka 5.

Tabulka 5: Procentuální zastoupení jednotlivých typů položek

	Vybavení	Hlášení	Zakázka
povinných	29 %	14 %	33 %
automatických	15 %	66 %	40 %
nepoužívaných	47 %	17 %	19 %
duplicitní	9 %	3 %	9 %

Zdroj: vlastní zpracování

Názorné porovnání jednotlivých typů položek je na grafu na Obrázek 17. Vzhledem k celkovému počtu položek je největší počet nepoužívaných polí na obrazovce Vybavení a nejvíce duplicitních položek obsahuje obrazovka Zakázka. Položky nepoužívané a duplicitní lze považovat za nadbytečné a položky povinné a automatické za nezbytné. Na základě tohoto tvrzení je vytvořena sumarizovaná Tabulka 6.



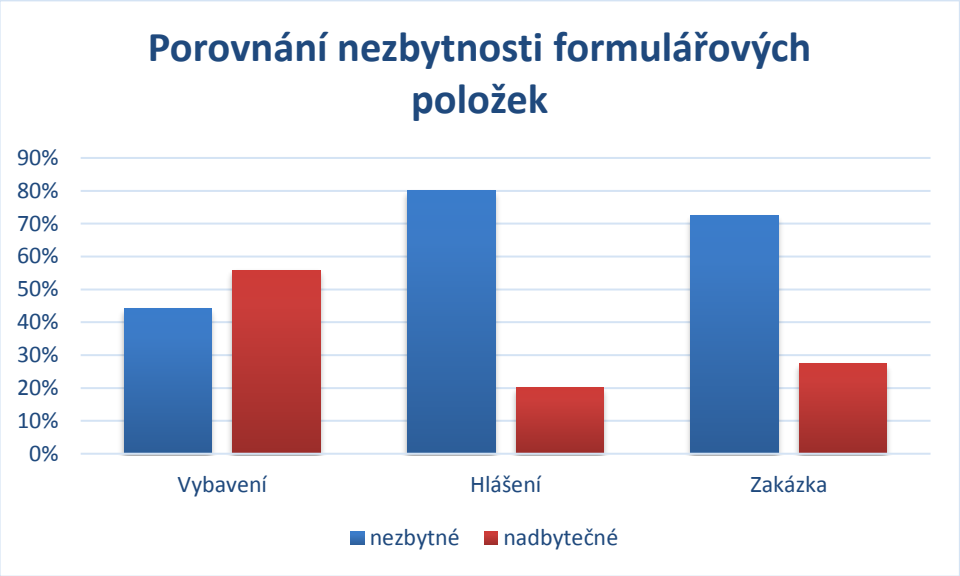
Obrázek 17: Položky formulářů

Zdroj: vlastní zpracování

Tabulka 6: Sumarizované položky

	Vybavení	Hlášení	Zakázka	Průměr
nezbytné	44 %	80 %	73 %	66 %
nadbytečné	56 %	20 %	28 %	34 %

Zdroj: vlastní zpracování



Obrázek 18: Porovnání nezbytnosti formulářových položek

Zdroj: vlastní zpracování

Ze sumarizované **Chyba! Nenalezen zdroj odkazů.** plyne, že nejvíce nadbytečných p oložek obsahuje formulář „Vybavení“, a to 56 %, lze tedy říci, že komplexita tohoto formuláře je optimální pouze z 44 %. Toto je způsobeno nedůslednou implementací systému, kdy nebyl formulář optimalizován a zbylo v něm 56 % nadbytečných polí, které neobsahují žádnou informaci. Formulář „Hlášení“ je z pohledu počtu položek optimální z 80 % a ze zkoumaných formulářů je nejlépe implementován. Formulář „Zakázka“ obsahuje 28 % nadbytečných polí, zde je velké zastoupení duplicitních položek, které je způsobeno tím, že formulář má 10 záložek, kdy na každé záložce jsou opakující se údaje. Tyto údaje by měly být v hlavičce formuláře, čímž by kles celkový počet položek i položek duplicitních. Veškerá nadbytečná formulářová pole zabírají místo v databázi a zhoršují přehlednost uživatelského rozhraní.

Z grafu na Obrázek 18 plyne, že v případě formuláře „Vybavení“ je počet nadbytečných polí vyšší než počet nezbytných. U tohoto formuláře je velké množství nepoužívaných polí. Důvod nepoužívání polí je, že uživatelé systému pro svoji práci nepotřebují uvádět tak detailní informace o vybavení, které jsou ve formuláři k dispozici. Toto je opět důsledek neoptimální implementace modulu PM systému SAP.

Výše provedená analýza uživatelského rozhraní se vztahuje na hodnocení provedené implementace hotového systému. Nadbytečná pole by se v systému nevyskytovala, kdyby byl systém vytvářen již od fáze návrhu. Překážkou pro snížení této komplexity je, že systém SAP je spravován externí IT firmou, kdy veškeré úpravy znamenají náklady navíc. Existuje, zde také hrozba ztráty dat a porušení konzistence databáze.

3.3.2. Multiplikace komplexity

V kapitole 2.5 bylo uvedeno, že přidáním 1 prvku do nějaké dimenze nedojde ke zvýšení komplexity pouze o jedna. Níže bude analyzován multiplikátor **K**, který bude ukazovat závislost přidání jednoho prvku do dané úrovně na celkovou komplexitu systému v dané fázi. Prvky budou přidávány do dimenze organizačního členění. Nejprve bude vypočten multiplikátor při přidání jednoho prvku organizační struktury v konceptuální úrovni ($\mathbf{K_k}$). Analogicky pak pro úroveň technologickou ($\mathbf{K_t}$), implementační ($\mathbf{K_i}$) a provozní ($\mathbf{K_p}$).

Po přidání 1 položky v podobě jednoho oddělení se 4 pracovníky bude modul PM navíc využívat další skupina uživatelů a zvýší se komplexita v konceptuální úrovni. Dojde k nárůstu počtu uživatelských účtů i rolí, dále k počtu hardwaru, počtu prvků databáze i počtu prvků uživatelského rozhraní. V konkrétním zkoumaném systému bylo zjištění, že toto navýšení způsobí nárůst komplexity o 33. Tuto hodnotu vzhledem k předpokladu linearity navyšování

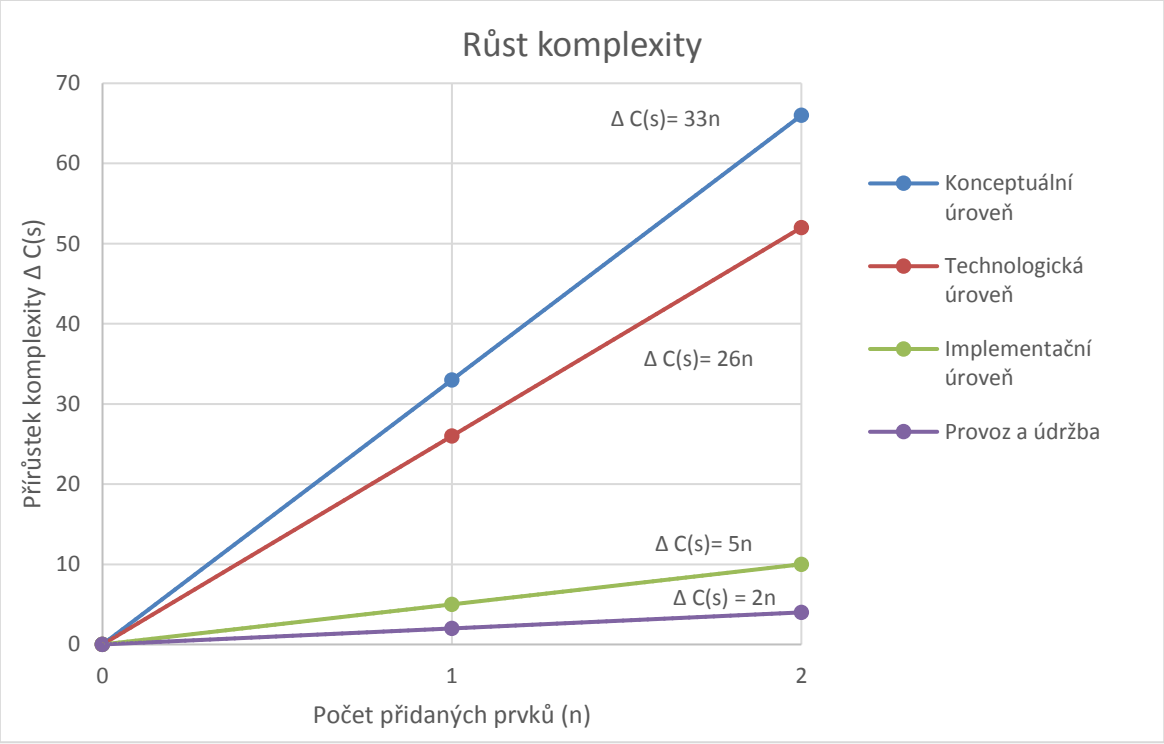
komplexity při přidávání prvků do organizační struktury považovat za hledaný multiplikátor $\mathbf{K_k}$.

Při přidání jednoho prvku v technologické úrovni, například přidáním dílčí plánovací skupiny o 2 členech, dojde k úpravám v databázi, dále k nárůstu uživatelských účtů, počtu uživatelských stanic a počtu prvků na obrazovkách systému. Komplexita zkoumaného systému se zvýší o 26. Tuto hodnotu lze označit za multiplikátor $\mathbf{K_t}$.

Pro získání multiplikátoru v implementační úrovni byla do existujícího systému přidána pouze jedna uživatelská role. Pro zajištění funkčnosti procesu bylo nutno zvýšit počet uživatelských oprávnění, hardwarových prvků i položek funkčních obrazovek. Nová uživatelská role také musela být zavedena do databáze. Bylo zjištěno, že došlo k přidání celkem 5 prvků, tedy $\mathbf{K_i} = 5$.

V provozní úrovni byl přidán pouze jeden uživatel do stávajících uživatelských skupin. V systému došlo k nárůstu pouze v seznamu uživatelů a uživatel byl vybaven uživatelskou stanicí. Multiplikátor $\mathbf{K_p}$ je tedy 2.

Růst komplexity při změnách v dimenzi organizačního členění v jednotlivých úrovních popisuje graf na Obrázek 19. Růst komplexity je v tomto případě lineární.



Obrázek 19: Růst komplexity

Zdroj: vlastní zpracování

V Tabulka 7 je znázorněn růst komplexity do jednotlivých obsahových dimenzích dle metodiky MMNDIS v každé úrovni abstrakce. Takto získané multiplikátory lze použít pro kvantifikaci komplexity při provádění změn v dimenzi týkající se organizačního členění. Obdobně by bylo možné definovat multiplikátory pro další obsahové dimenze. Ze získaných výsledků je patrné, že multiplikátor postupně klesá, což souvisí s úrovní abstrakce dané změny.

Větších hodnot multiplikátoru by bylo získáno například při rozšíření podnikové strategie a zapracování nových procesů do systému. V takovém případě by bylo nutno zvážit nákladnost implementace nových procesů do stávajícího systému v porovnání se systémem novým.

Tabulka 7: Růst komplexity v jednotlivých obsahových dimenzích

	Data/ informace	Funkce/ procesy	Organizační členění	Software	Hardware	Uživatelské rozhraní	Δ C (s), n=1
Konceptuální úroveň	Diagram tříd	Diagram aktivit	Organizační struktura	Diagram komponent	Diagram nasazení	Use case diagram	33
Technologická úroveň	Relační databáze	Eriksson-Penker notace	Organizační struktura	Transakce a funkce	Topologie sítě	GUI modely	26
Implementační úroveň	Databáze	BPMN	Uživatelské role	SLOC	Počet konfiguračních položek	Počet obrazovek	5
Provoz a údržba	Databáze	BPMN	Uživatelská oprávnění	SLOC	Počet konfiguračních položek	Počet položek obrazovek	2

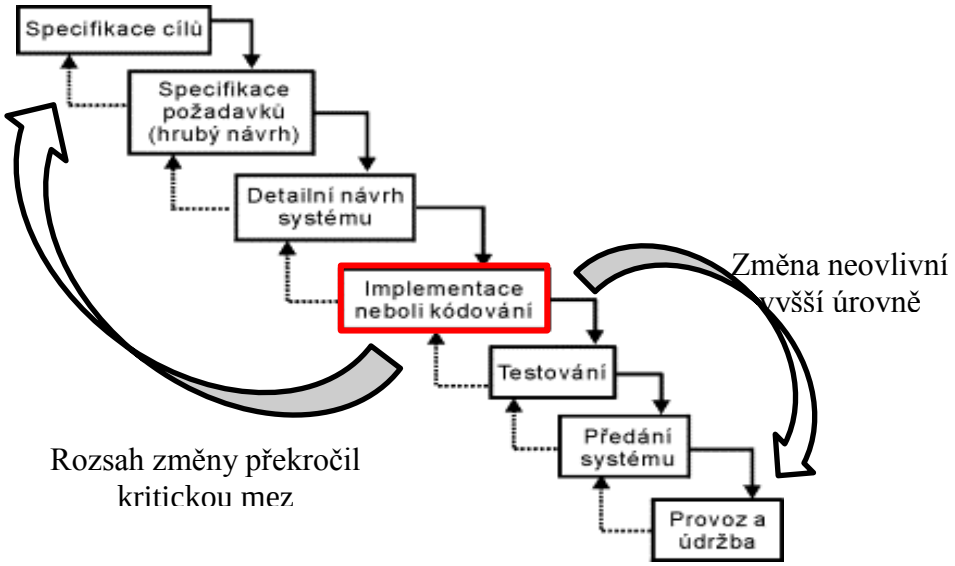
Zdroj: vlastní zpracování

V Tabulka 7 je naznačeno, jak se projeví provedení změny v jednotlivých úrovních. Provedenou analýzu lze označit za horizontální. Obdobně by bylo možné provézt analýzu změny komplexity vertikálním směrem, kde by byl zkoumán multiplikátor dimenzí při přidání

nových prvků. Pro určení těchto multiplikátorů jsou nutné detailní informace o zkoumaném systému, tyto však s ohledem na to, že je systém SAP ve správě externí firmy, nejsou dostupné.

Vždy je důležité při zkoumání komplexity jednotlivých dimenzí vycházet z použité architektury životního cyklu. Spirálový model uvedený v kapitole 1.4.3 přechází od prototypu opět k úpravě návrhu. Bylo by tedy vhodnější měřit komplexitu jednotlivých prototypů nikoliv fáze abstrakce. Obdobně toto platí i pro obsahové dimenze. Změna komplexity mezi jednotlivými prototypy by pak odpovídala změně komplexity obsahových dimenzí. Při použití spirálové architektury životního cyklu má každá požadovaná změna za následek kontrolu požadavků a verifikaci návrhu celého systému.

Při použití vodopádového modelu z kapitoly 1.4.1 by bylo možno kvantifikovat počet úrovní, ve kterých se projeví daná změna. Opět by byl potvrzen fakt, že změna na vyšší úrovni má zásadní dopad na úrovně nižší.



Zdroj: vlastní zpracování

Obrázek 20: Vliv změny komplexity ve vodopádovém modelu

V rámci vodopádového životního cyklu je možno zkoumat, jak požadovaná změna ovlivní jednotlivé fáze, tedy o kolik stupňů je třeba vystoupat výš směrem k návrhu, aby bylo požadavkům vyhověno. Například lze určit, zda libovolná požadovaná změna (nové prvky a vazby) ve fázi implementace bude mít za následek pouze nárůst velikosti databáze nebo bude nutné změnit celý návrh systému. Tímto způsobem je možné nalézt kritickou hranici změny komplexity, kdy si požadovaná změna systému vyžádá kompletní reengineering. Toto ilustruje Obrázek 20.

Postup kvantifikace komplexity informačního systému v jednotlivých fázích a obsahových dimenzích je tedy závislý na použité architektuře životního cyklu.

ZÁVĚR

Práce byla věnována komplexitě informačních systému. Byly vymezeny jednotlivé typy informačních systému a byl definován pojem životní cyklus a jeho používané modely. V rámci životního cyklu byly objasněny druhy zavádění systému, což je velmi důležité ve vztahu ke komplexitě, neboť právě komplexita má hlavní vliv na volbu postupu zavádění nového informačního systému.

Byla rozlišena komplexita nezbytná a nadbytečná. Tyto dva druhy komplexity mají zásadní vliv na vývoj informačního systému, neboť snahou by mělo být vždy nadbytečnou komplexitu systému nezanášet do infomačního systému. Dále byla posouzena komplexita na všech úrovních abstrakce v rámci vývoje informačního systému včetně dopadů na organizaci. Lze sumarizovat, že příliš vysoká komplexita má za následek vždy růst nákladů a neefektivní práci, proto je třeba komplexitu sledovat v rámci celého životního cyklu a nenavyšovat nadbytečnou komplexitu, ale zároveň vytvořit systém dostatečně složitý, aby obsáhl všechny aspekty daného procesu, který má být v nejlepším případě automatizován. Byl vytvořen postup pro kvantifikaci komplexity v jednotlivých fázích životního cyklu dle metodiky MMNDIS založený na počtu prvků a vazeb v jednotlivých dimenzích.

V poslední kapitole byl analyzován modul PM systému SAP ve vybrané společnosti. Byl popsán proces, který je pomocí systému zpracováván, a byly identifikovány jednotlivé atributy entit, které v tomto procesu figurují. Při posuzování komplexity uživatelského rozhraní bylo zjištěno, že formuláře obsahují průměrně 34 % nadbytečných položek, které způsobují nadbytečnou komplexitu, která ztěžuje uživateli práci. Nadbytečná pole můžou být důsledkem tvorby „rezervy“, kdy v okamžiku implementace nebylo jasné, zda bude položka využívána, ale byla ponechána viditelná s ohledem na její možné využití v budoucnosti. Hlavní příčinou tohoto může však být zejména nedokonalá implementace tohoto modulu, kdy systém SAP nabízí velké množství položek, které je třeba protřídit a rozhodnout o jejich použitelnosti v procesu a nepotřebné skrýt z obrazovek. Nadbytečná a redundantní pole jsou také důsledkem toho, že bylo implementováno hotové systémové řešení, které nebylo vytvořeno přesně na míru procesu, který má podporovat. Dále byl na základě postupů z kapitoly 2 zjištěn multiplikátor komplexity tohoto systému při provádění změn v organizační struktuře. Výsledkem je, že multiplikátor je největší v konceptuální úrovni a v dalších úrovních se snižuje. Multiplikátor v konceptuální úrovni je cca 6x větší než v úrovni implementační.

Závěrem lze říci, že všech cílů práce bylo dosaženo. Práci by bylo možno rozšířit využitím jiných kvantifikačních metod například pomocí entropie nebo logické hloubky.

POUŽITÁ LITERATURA

[1] Activity diagram - diagram aktivit. *OO, UML, analýza, metodologie* [online]. 2005 [cit. 2018-03-06]. Dostupné z: http://mpavus.wz.cz/uml/uml-b-activity-3-2-3.php

[2] BERÁNEK, Ladislav. Vytváření webových stránek pro e-comerce. Podnikání a obchodování na internetu [online]. Jihočeská univerzita v Českých Budějovicích, 2010 [cit. 2018-03-06]. Dostupné z: http://ecom.ef.jcu.cz/web/download/teorie/p04-vytvareni_webovych_stranek.pdf

[3] Business Process Modeling Notation (BPMN) Information. *Object management group* [online]. 2018 [cit. 2018-03-06]. Dostupné z: http://www.omg.org/bpmn/Documents/FAQ.htm

[4] Component diagram - diagram komponent. *OO, UML, analýza, metodologie* [online]. 2005 [cit. 2018-03-06]. Dostupné z: http://mpavus.wz.cz/uml/uml-s-component-3-3-2.php

[5] ČAPKA, David. UML - Class diagram. *ITnetwork.cz* [online]. 2013 [cit. 2018-03-06]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-class-diagram-tridni-model

[6] ČÁPKA, David. Úvod do UML. *ITnetwork* [online]. 2013 [cit. 2018-03-06]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-uvod-historie-vyznam-a-diagramy

[7] Druhy informačních systémů. *Prostředky informačních technologií* [online]. Masarykova univerzita, 2002 [cit. 2018-03-06]. Dostupné z: http://pit.wz.cz/informacni-systemy.php

[8] Eriksson-Penker Extensions. *Sparx Systems* [online]. 2000 [cit. 2018-03-06]. Dostupné z: http://www.sparxsystems.com/enterprise_architect_user_guide/10/domain_based_models/eriksson_penker_extensions.html

[9] *Globální architektura IS/IT* [online]. Mendelova univerzita v Brně [cit. 2018-03-06]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=4756

[10] HOLUB, I. Jak a proč vzniká komplexita v IS. Systémová integrace. Praha: Česká společnost pro systémovou integraci, 2012, 1, s. 74-93. ISSN: 1804-2716 (online)

[11] HOLUB IT CONSULTING. Komplexní systémy. holub.cz [online]. [cit. 2017-03-03]. Dostupné z: http://www.holub.cz/index.php/sk/komplexni-systemy.

[12] Informační systémy: Základní pojmy a souvislosti. *Dokumentační portál Centra informačních technologií VŠB-TUO* [online]. Ostrava: VŠB-TUO Ostrava, 2016 [cit. 2018-03-07]. Dostupné z: http://homen.vsb.cz/~s1i95/isvdas/is/is_uvod.htm

[13] JANÍČEK, Přemysl a Jiří MAREK. *Expertní inženýrství v systémovém pojetí*. Praha: Grada, 2013. Expert (Grada). ISBN 978-80-247-4127-7.

[14] Konfigurační položky. *IBM Knowledge Center* [online]. USA: IBM Corporation, 2018 [cit. 2018-03-06]. Dostupné z: https://www.ibm.com/support/knowledgecenter/cs/SSANHD_7.5.1/com.ibm.sccd.doc/config/c_config_item.html

[15] Large scale management of distributed systems 17th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2006, Dublin, Ireland, October 23-25, 2006: proceedings. Berlin: Springer, 2006, s. 173-184. ISBN 9783540476627.

[16] MALEČKOVÁ, D. 2013. Informační věda a teorie komplexity. Univerzita Karlova v Praze, 11–15.

[17] MANAGEMENT MANIA. Podnikový proces. [online]. [cit. 2017-03-03]. Dostupné z: https://managementmania.com/cs/business-process-podnikovy-proces.

[18] MANAGEMENT, MARKETING. Vše co student potřebuje vědět. Komplexita se vztahuje. studentske.eu [online]. [cit. 2017-03-03]. Dostupné z: http://managment-marketing.studentske.eu/2008/06/komplexita-se-vztahuje.html

[19] MANAGEMENT MANIA. *MMDIS (Multidimensional Management and Development of Information Systems)* [online]. 2016 [cit. 2018-03-06]. Dostupné z: https://managementmania.com/cs/mmdis

[20] MOLNÁR, Zdeněk. *Moderní metody řízení informačních systémů*. Praha: Grada, 1992. ISBN 8085623072.

[21] Multidimesional Management and Development of Information Systém (MMDIS). *MBI (Management Byznys Informatiky)* [online]. 2014 [cit. 2018-03-06]. Dostupné z: http://mbi.vse.cz/public/cs/obj/METHOD-68

[22] Organization Chart. *Google developers* [online]. 2018 [cit. 2018-03-06]. Dostupné z: https://developers.google.com/chart/interactive/docs/gallery/orgchart

[23] Organizační struktura (Organizational Structure). *Management mania* [online]. 2016 [cit. 2018-03-06]. Dostupné z: https://managementmania.com/cs/formalni-organizacni-struktura

[24] ŘEPA, Václav. *Modelování business systémů* [online]. KIT VŠE Praha [cit. 2018-03-05]. Dostupné z: https://www.researchgate.net/profile/Vaclav_Repa3/publication/228837378_Modelovani_business_systemu

[25] ŘEPA, Václav. *Podnikové procesy: procesní řízení a modelování*. 2., aktualiz. a rozš. vyd. Praha: Grada, 2007, s. 179-180. ISBN 8024722526.

[26] Source lines of code. *ProjectCodeMeter* [online]. 2018 [cit. 2018-03-06]. Dostupné z: http://www.projectcodemeter.com/cost_estimation/help/GL_sloc.htm

[27] SPRAGUE, Ralph H. A Framework for the Development of Decision Support Systems. *MIS Quarterly* [online]. 1980, **4**(4), 1- [cit. 2018-03-06]. DOI: 10.2307/248957. ISSN 02767783. Dostupné z: http://www.jstor.org/stable/248957?origin=crossref

[28] ŠMÍD, Vladimír. Pojem informačního systému. *Management informačního systému* [online]. Masarykova univerzita: Fakulta informatiky [cit. 2018-03-06]. Dostupné z: https://www.fi.muni.cz/~smid/mis-infsys.htm

[29] TVRDÍKOVÁ, Milena. EIS - nezbytná součást business intelligence. *Systém online* [online]. Systém online, 2002 [cit. 2018-03-06]. Dostupné z: https://www.systemonline.cz/clanky/eis-nezbytna-soucast-business-intelligence.htm

[30] Use case diagram - diagram případů užití. *OO, UML, analýza, metodologie* [online]. 2005 [cit. 2018-03-06]. Dostupné z: http://mpavus.wz.cz/uml/uml-b-use-case-3-2-1.php

[31] WHITE, Stephen A. Process Modeling Notations and Workflow Patterns. *Workflow Patterns* [online]. USA: Workflow Patterns Initiative, 2017 [cit. 2018-03-06]. Dostupné z: http://www.workflowpatterns.com/vendors/documentation/BPMN_wfh.pdf

SEZNAM PŘÍLOH

Příloha A: Proces realizace opravy