ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA ELEKTROTECHNICKÁ

KATEDRA TECHNOLOGIÍ A MĚŘENÍ

DIPLOMOVÁ PRÁCE

Metody a nástroje pro generování testů pro HiL testování

Vedoucí práce: Ing. Michal Kubík, Ph.D.

Autor práce: Bc. Veronika Housarová

Plzeň 2012

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2011/2012**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Veronika HOUSAROVÁ**
Osobní číslo:	**E10N0021P**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Komerční elektrotechnika**
Název tématu:	**Metody a nástroje pro generování testů pro HiL testování**
Zadávající katedra:	**Katedra technologií a měření**

Zásady pro vypracování:

1. Seznamte se s principy metod použitelných pro generování testů při HiL testování.
2. Pro jednotlivé metody vyberte vhodné softwarové nástroje pro podporu generování testů s ohledem na pořizovací náklady.
3. Pomocí vybraných nástrojů vygenerujte testy pro dané testovací případy.

Rozsah grafických prací:	**podle doporučení vedoucího**
Rozsah pracovní zprávy:	**30 - 40 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**
Seznam odborné literatury:	

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí diplomové práce:	**Ing. Michal Kubík** Katedra aplikované elektroniky a telekomunikací
Datum zadání diplomové práce:	**17. října 2011**
Termín odevzdání diplomové práce:	**11. května 2012**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Ing. Vlastimil Skočil, CSc.
vedoucí katedry

V Plzni dne 17. října 2011

Anotace

Práce se zabývá principy metod použitelných pro generování testů při HIL testování. V úvodní části je popsáno, jak důležité je testování, V-model, testovací proces a princip HIL testování. V další části jsou sepsány metody pro specifikaci testu jak formální techniky tak i neformální techniky. V poslední části je popsána základní verze programu CTE XL a je zde ukázka vygenerovaného testu.

Klíčová slova

Testovací proces, specifikace testu, MCDC, klasifikační strom, CTE XL

Methods and software tools for generating test cases during Hil testing

Abstract

This thesis deals with principles and methods for generating tests for hardware in the loop testing (HiL). In the introduction the importance of testing, a V-model of the development and testing processes and the principles of HiL testing are mentioned. This is followed by the description of the methods for test specification. Both formal and informal techniques are included. In the last section the basic version of the CTE XL (Classification Tree Editor eXtended Logics) is described and in this program is generated example of the test case.

Key words

Test process, Test specification, MCDC, Classification Tree, CTE XL

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

Dále prohlašuji, že všechny informace obsažené v této diplomové práci jsou volně publikovatelné a nepodléhají zvláštnímu utajení.

Dále prohlašuji, že veškerý software použitý při řešení této diplomové práce je legální.

V Plzni dne 9.5.2012 Bc. Veronika Housarová

……..

Poděkování

Na tomto místě bych ráda poděkovala Ing. Stanislavu Veverkovi ze společnosti MBtech Bohemia s.r.o. za jeho čas a úsilí, které mi v průběhu vytváření této práce věnoval, bez jeho cenných rad a připomínek by moje práce v této podobě nemohla vzniknout. Dále bych chtěla poděkovat vedoucímu mé práce Ing. Michalu Kubíkovi z Katedry aplikované elektroniky a telekomunikací, ZČU v Plzni, jehož přínos byl taktéž velmi významný. V neposlední řadě také děkuji mé rodině za poskytnutou podporu a zázemí během mého studia a při tvorbě této práce.

Obsah

Seznam symbolů a zkratek

Anglické zkratky

Zkratka	Význam zkratky
CAN	Controller Area Network – Síť v oblasti řídících jednotek
CTE XL	Classification Tree Editor eXtended Logics – Editor pro tvorbu klasifikačních stromů
CTM	Classification Tree Method – Metoda klasifikačního stromu
ECU	Electronic Control Unit – Elektronická řídící jednotka
FIU	Failure Insertion Unit – Jednotka pro vkládání chyb
HiL	Hardware in a Loop - Hardware ve smyčce
IEC	International Electrotechnical Commission – Mezinárodní elektrotechnická komise
ISO	Industry Standard Organization – Organizace pro průmyslovou standardizaci
IT	Information technology – Informační technologie
MCDC	Modified Condition Decision Coverage - Modifikované pokrytí podmínky/rozhodnutí
PC	Personal Computer – Osobní počítač
SPICE	Simulation Program with Integrated Circuit Emphasis - Simulační program s důrazem na integrované obvody

České zkratky

Zkratka	Význam zkratky
MOT	Motocykl
NA	Nákladní automobil
NZŘ	Národní zkušební řád
OA	Osobní automobil
SUV	Sportovní užitkové vozidlo

1 Úvod

V současné době je elektronika v automobilech nedílnou součástí našeho života. Tato elektronika se stále vyvíjí a je dokonalejší a dokonalejší, a to jak u nejvyšší třídy, tak i u nejnižších tříd. Ale měli bychom si položit otázku: „Je tato technika bezpečná?“ V dnešní době je testování velice důležité. Pokud bychom netestovali, mohlo by to mít velice nepříznivé následky.

V 80. letech minulého století stačilo, aby automobil startoval, troubil, stíral okna, svítil, blikal, topil resp. větral. Vrcholem elektroniky v té době (v tehdejším Československu) bylo autorádio a elektronické zapalování. Je nutno zmínit, že i v této době již existovaly automobily s mnohem vyšší elektronickou výbavou. Jako příklad si ukažme Mercedes-Benz W123, který se vyráběl od roku 1975 a patřil mezi špičku tehdejších automobilů. V jeho výbavě se nacházelo např. elektrické stahování oken, elektrické ovládání zrcátek, centrální zamykání, apod. Tak jako byla nesrovnatelná vybavenost a komfortnost vozů Škoda 120 a Mercedes-Benz W123, byla podobně nesrovnatelná i pořizovací cena.

V dnešní době trh neustále vyvíjí tlak na zvyšování kvality a vybavenosti i u automobilů nejnižší třídy, což však přináší zvyšování ceny a samozřejmě i zvyšování bezpečnosti a spolehlivosti. Se spolehlivostí souvisí obory jako testování a diagnostika. V poslední době se ve větší míře začíná věnovat pozornost testování i elektronických systémů ve vozidle. Dříve se testování provedlo jednoduše, stylem svítí-nesvítí, stírá-nestírá, atd. Dnes je situace mnohem složitější. Vzhledem ke složitosti elektronického systému dnešního automobilu, je nutné přistupovat k testování komplexně již během celého procesu vývoje automobilu.

V následujících kapitolách jsou popsány metody pro specifikaci testu, které jsou důležité pro tvorbu testovacích případů a je zde vygenerován ukázkový test ve volně stažitelné základní verzi nástroje CTE XL od společnosti Berner & Mattner.

2 Testování

2.1 Pojem testování

V dnešním světě považujeme testování za jednu z velice důležitých věcí našeho života, ať už testování našich znalostí ve škole, se kterým se setkal snad každý z nás, nebo testování elektronických součástek (správné funkčnosti elektronického zařízení). Všichni jsme se setkali nebo ještě setkáváme s testováním ve škole formou zkoušení. Slovo test u většiny lidí vyvolává spíše nepříjemný pocit a možná proto si pokládáme otázku: „Je testování důležité?“

Přibližme si testování elektronických zařízení na oboru kynologie (cvičení a zkoušení psů). Psovod se rozhodne, na jaký druh zkoušky bude trénovat a kterou zkoušku chce vykonat. V národním zkušebním řádu (NZŘ) je uveden seznam zkoušek (co testovat) a každá zkouška je zde důkladně popsána (specifikace testu). Poté následuje zkoušení (samotný test). Rozhodčí sleduje výkon psovoda se psem a zpracovává tyto výkony (odezvy) nejčastěji porovnáváním výkonů se správnými (podle NZŘ). Kvalita testovacího procesu závisí na tom, jak provedené zkoušky pokrývají obsah národního zkušebního řádu. Podobné je to u elektronických systémů. Je zapotřebí znát specifikaci testovaného objektu, resp. požadavky na něj (podobně jako popis zkoušky v NZŘ) a poté je možné navrhnout vhodný test, aby mohlo být zaručeno splnění specifikace (požadavků), v případě že testovaný objekt dává požadované odezvy.

Počet úkonů musí být omezen z toho důvodu, že zkoušení psovoda se psem nemůže být nekonečné. Mělo by se za co nejkratší dobu zjistit, zda je psovod schopen tyto úkony vykonat. Můžeme předpokládat, že v průběhu zkoušení dojde k výskytu typických chyb, kterých se psovodi se psem velmi pravděpodobně dopustí. Úkony by měly být navrženy tak, aby odhalily tyto chyby, a i když výkony psovodů jsou správné, je nutné připustit určitou prospěchovou toleranci v rámci daného chybového modelu. Při testování elektronických systémů se využívá modelování poruch. Je-li model poruch úspěšně vyzkoušen, přisuzuje se mu věrohodnost. Lze očekávat, že systém bude spolehlivý, pokud bude testováno velké, resp. požadované procento modelových chyb.

V případě, že psovod danou zkouškou neprojde, musí zkoušku opakovat. Tento postup se shoduje s testováním elektronických systémů. Psovod si může usnadnit práci, když si dopředu zjistí informace o obsahu zkoušky – z čeho se bude zkoušet, a pak si může přípravu lépe naplánovat a následně uspět.

Význam a cíle testování v automobilech

V průběhu celého vývoje řídících jednotek v automobilu probíhají rozsáhlé testy. Samozřejmostí je i samotné otestování konečného produktu. Cílem testování vozidel je, aby byla dosažena určitá úroveň konečného produktu (automobilu). Zároveň k tomu patří také ověření posledních vývojových kroků a v neposlední řadě také ověření vlastností vozidla vzhledem k očekávání zákazníka [1].

2.2 Bezpečnostní normy a standardy

Vzhledem k velkému počtu řídících jednotek (ECU) v automobilu je zapotřebí dodržovat bezpečností normy a standardy. Je nutné otestovat všechna zařízení tak, aby odpovídala těmto bezpečnostním normám.

V evropském automobilovém průmyslu není významnější standard řízení procesů a kvality vývoje software než Automotive SPICE™. Výsledky auditu dodržování tohoto standardu jsou rozhodujícím kritériem většiny automobilových společností při výběru svých partnerů. Význam standardu výrazně roste v poslední době zejména s rostoucím objemem softwaru při výrobě i v rámci samotných automobilů. Zásadním přelomem se stalo nasazování softwaru i v životně důležitých funkcích vozidel včetně řízení a brzd. Od roku 2005 se standard Automotive SPICE™ stal oborovou variantou oficiálního standardu ISO/IEC 15504, jehož postupy upřesňuje pro podmínky automobilového průmyslu. Standard Automotive SPICE™ podporuje řada významných evropských automobilových koncernů: AUDI AG, BMW Group, Daimler AG, Fiat Auto S.p.A., Ford Werke GmbH, Jaguar, Land Rover, Porsche AG, Volkswagen AG a Volvo Car Corporation. Společnosti se sdružily do zájmové skupiny SPICE User Group, která o rozvoj standardu pečuje. Díky této podpoře je standard téměř každoročně upravován, aby odpovídal aktuálním poznatkům a zkušenostem s vedením projektů [2].

Normy pro bezpečnost se mohou lišit podle toho, v jaké zemi bude vozidlo provozováno. Tyto normy záleží na požadavcích daného státu. Aby bylo možné vozidlo v daném státě provozovat, stát vydá tzv. homologaci. Homologací se rozumí ověření vlastností vozidla z hlediska přípustnosti jeho použití. Vozidla musí splňovat státem dané normy. V případě, že tyto normy nesplňují, stát nevydá homologaci a to znamená, že tento model nemůže být provozován v daném státu. Tyto homologace nejsou tak přísné, jak by si některé společnosti představovaly a z toho důvodu zavádí vlastní interní normy, které následně dodržují.

Pro silniční vozidla se vztahuje řada norem ISO DIS 26262. ISO je zkratka pro International Organization for Standardization (Mezinárodní organizace pro normalizaci). Tato mezinárodní síť organizací se sídlem v Ženevě koordinuje uspořádání a publikování schválených norem. V červenci roku 2009 byla publikována norma ISO/DIS 26262, která popisuje současný stav v rozvoji příslušných bezpečnostních funkcí vozidla. Výchozím bodem pro všechny bezpečnostní činnosti v souladu s ISO 26262 je analýza rizik pozorované funkce. Na základě této analýzy je možné definovat návrhy zabezpečení a získávat technické bezpečnostní koncepty pro konkrétní systém. Prakticky to znamená, že se zaměříme na rozdělení funkcí. Při odvozování bezpečnostních opatření se počítá i s určitým stupněm volnosti, který můžeme využít, aby při řešení bezpečnostního návrhu byly náklady co nejnižší. Případná chyba systémového komponentu nesmí vést k nebezpečnému selhání. Poté můžeme definovat, které moduly jsou bezpečnostně relevantní a které bezpečnostní stupně musíme dosáhnout. Požadavky na modul vývoje a testovací metody jsou popsány v ISO 26262. Norma ISO 26262 je placená. Cena této normy není vysoká, pohybuje se okolo 2 500 Kč. Dražším elementem jsou odborníci, kteří se v této problematice vyznají a jsou schopni tuto normu aplikovat na konkrétní případy. Více informací lze nalézt v [3].

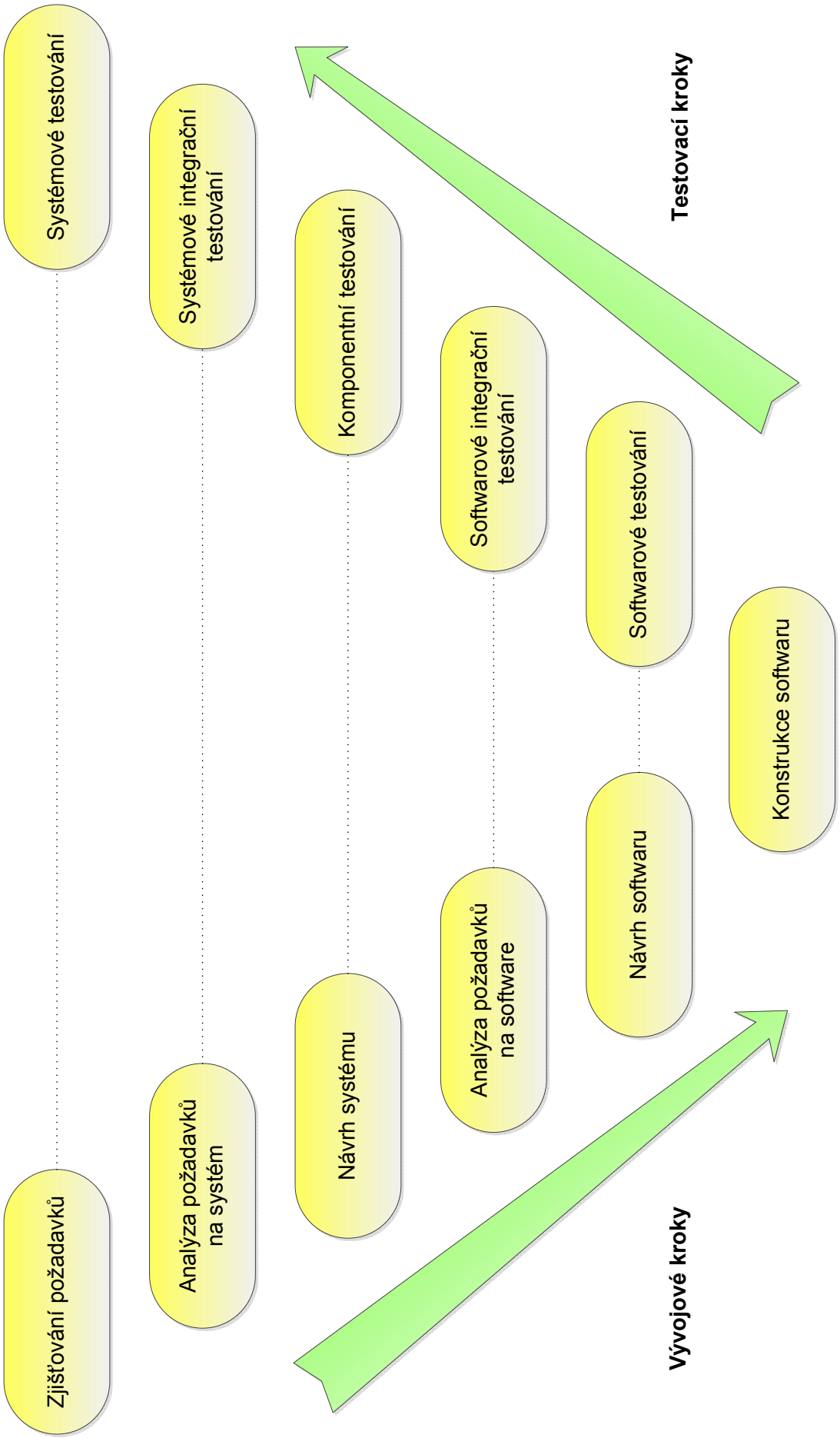
ISO 26262 je používána v souvislosti s bezpečnostními systémy, které obsahují jeden nebo více elektrických / elektronických (E/E) systémů a tyto systémy jsou následně instalovány v sériových výrobních vozidlech s maximální technicky přístupnou hmotností do 3 500 kg. ISO 26262 se nezabývá E/E systémy ve zvláštních účelových vozidlech jako jsou například vozidla určená pro řidiče se zdravotním postižením. ISO 26262 se týká možných rizik způsobených poruchou chování E/E bezpečnostních systémů, včetně vzájemného působení těchto systémů.

Nezabývá se nebezpečím souvisejícím s elektrickým proudem, požárem, kouřem, radiací, toxicity, hořlavosti, korozi a podobném nebezpečí, pokud není přímo způsobeno poruchou chování E/E bezpečnostních systémů [3].

2.3 V – model vývojového procesu

V současné době je V-model často používaným a doporučeným procesem. Jedná se o vývojový standard, který přichází z IT-odvětví a byl vyvinut z vodopádového modelu. Vodopádový model (angl.: waterfall) je nejjednodušším modelem využívaným k popisu procesu vývoje. Tento původní model je přímočarý (lineární), kdy jednotlivé etapy navazují jedna na druhou s minimálním časovým překrytím, a v jeden okamžik se v rámci jednoho projektu pracuje na právě jedné etapě. V-model je v místě implementace zalomen tak, že tvoří písmeno „V“. To umožňuje zlepšit účinnost testování a snížit náklady na odstranění chyb. Tento model je v procesu vývoje automobilových řídicích systémů velmi aktuální.

Na levé straně V-modelu jsou naznačeny vývojové kroky, které jsou následně na pravé straně testovány. Interpretace záleží na vlastní roli procesu, tedy pohledu na celek. V této souvislosti je pod pojmem celek vyznačován systém, kterým může být například veškeré elektrické a elektronické vybavení ve vozidle. Zároveň se ale může jednat jen o distribuovanou funkci jako je jízdní asistenční systém nebo například samostatná řídící jednotka. Základní myšlenkou V-modelu je minimalizace nákladů na odstranění chyb. Z tohoto důvodu se testování provádí paralelně s vývojovou částí, aby bylo dosaženo co nejnižších nákladů [1].



Obr.č. 2.1 Znázornění V-modelu vývojového procesu

Jednotlivé vývojové kroky s ohledem na proces z automobilového standardu SPICE^{TM} znázorněné na obr.č. 2.1 jsou následující:

- **Zjišťování požadavků** *(angl. - Requirements elicitation)*

Jsou zde shrnuty zákaznické potřeby a požadavky a slouží jako základ pro definici požadovaného produktu. Požadavky a potřeby jsou definovány uživatelem nebo vývojovým inženýrem daného systému.

- **Analýza požadavků na systém** *(angl. – System requirements analysis)*

V této fázi se koná:

- popis požadavků, ze kterých má být vyvinut systém a jejich technické a organizační řešení,
- riziková analýza (management rizik),
- zpracování technického modelu pro funkci, data a objetky, tzn. zde jsou shrnuty požadavky do funkcionality celého auta, které poté slouží jako podklady pro systém – design.

- **Návrh systému** *(angl. – System design)*

V tomto kroku je provedeno rozložení systémů do jednotlivých segmentů, které mají rozdílné systémové požadavky (jako příklad typického systému můžeme uvést venkovní osvětlení).

- **Analýza požadavků na software** *(angl. – System architectural design)*

Jedná se o zjištění přesných nároků na software. Požadavky na software by měly být proveditelné, měřitelné, testovatelné a také by měly být dostatečně detailně popsané.

- **Návrh softwaru** *(angl. – Software design)*

Nejprve jsou navrhnuty interní a externí rozhraní a poté je specifikována integrace. V detailním rozboru jsou pak popsány jednotlivé komponenty, moduly a data.

- **Konstrukce softwaru** *(angl. – Software construction)*

V předchozích fázích již byly specifikovány moduly a algoritmy a zde jsou následně převedeny do kódu.

- **Softwarové testování** *(angl. – Software unit testing)*

V této fázi jsou testovány jednotlivé spustitelné softwarové moduly. Testuje se zde především tzv. průchodnost kódu.

- **Softwarové integrační testování** *(angl. – Software integration testing)*

Během této fáze jsou integrovány jednotlivé sortwarové moduly za použití definovaných rozhraní. Tento software je následně testován nezávisle na hardwarové platformě.

- **Komponentní testování** *(angl. – Component testing)*

V této fázi jde o integraci různých softwarových a hardwarových jednotek. Tato integrace vede ke kompletnímu systému - v automobilových sítích k řídící jednotce, která je testována jako komponenta.

- **Systémové integrační testování** *(angl. – System integration testing)*

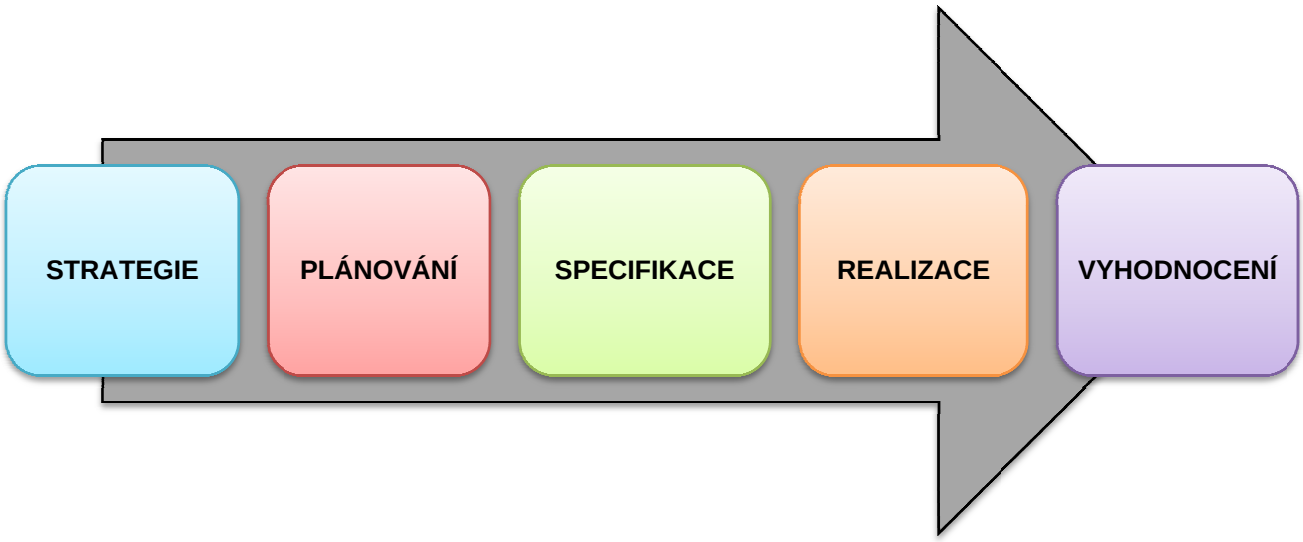
Jedná se o integraci více jednotek do celkové architektury vozidla. Po integraci řídících jednotek do systému vzniká síť řídících jednotek v automobilu, z čehož plyne integrační testování, tzn. test celkového systému proti základním požadavkům systému.

- **Systémové testování** *(angl. – System testing)*

V této fázi testování se všechny řídící jednotky nachází v reálném prostředí. Systém je testován jako celek (např. testování vozidla) [1].

2.4 Testovací proces

K realizaci testovací kampaně lze aplikovat tzv. testovací proces, který je rozdělen do 5 kroků. Je možné použít více či méně kroků testovacího procesu, záleží na požadavku zákazníka. V této práci je popsán testovací proces, který se skládá ze strategie, plánování, specifikace, realizace a vyhodnocení. Jednotlivé části testovacího procesu na sebe navazují a neměly by být jakkoli zaměňovány nebo přeskakovány. Na začátku celého procesu může být požadavek zákazníka, který například stanoví objekt samotného testování, rozsah, termín dokončení a další parametry. V následujících kapitolách (2.4.1 – 2.4.5) jsou detailně popsány jednotlivé části tohoto procesu. Testovací proces je graficky znázorněn na obr.č. 2.2.

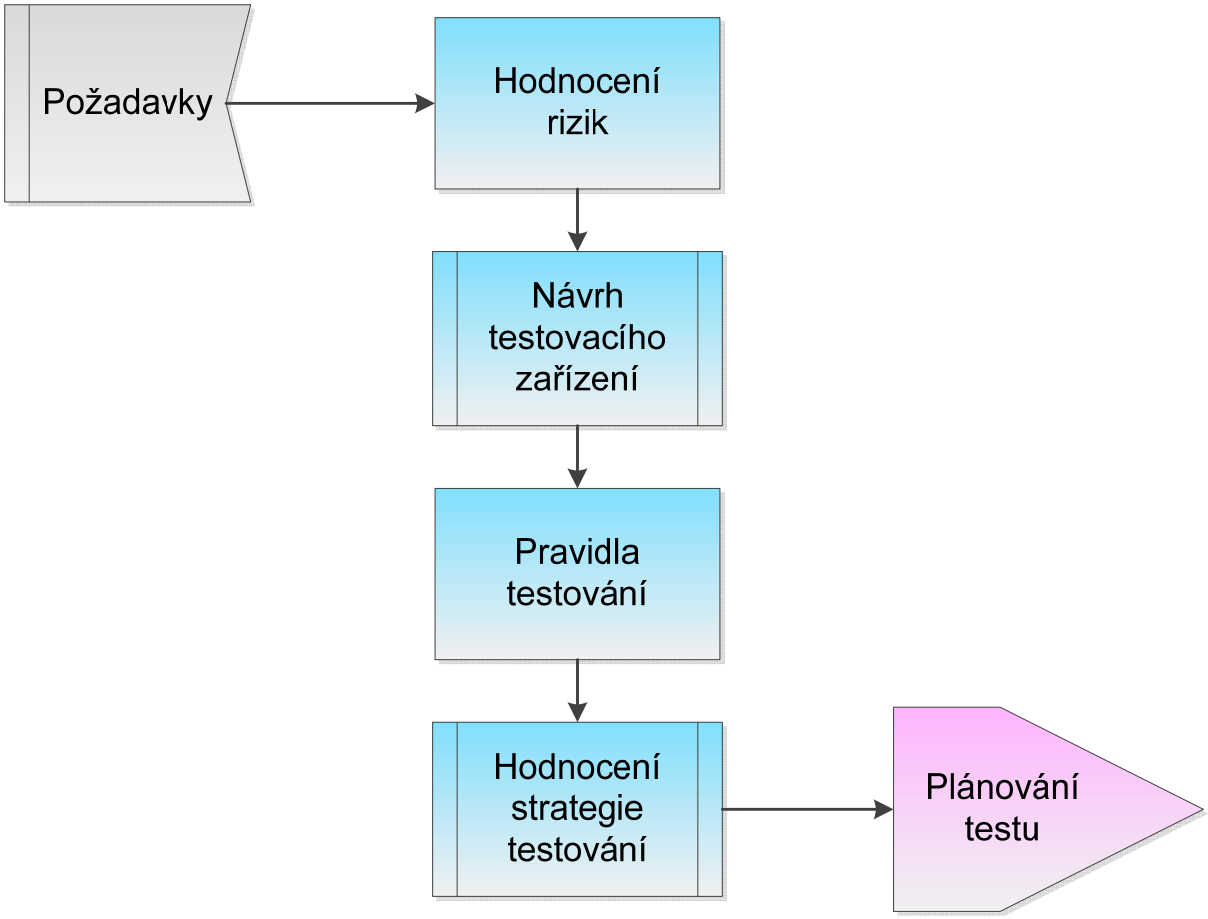


Obr.č. 2.2 Testovací proces

2.4.1 Strategie testovací kampaně

Ověřeným prvním krokem testovacího procesu je tzv. navržení strategie testování. Hlavním cílem této části procesu je, aby riziko neodhalené chyby při zahájení výroby bylo co nejnižší a využití prostředků (zdrojů) bylo co nejefektivnější. Zdroje pro testovací proces jsou peníze, lidé a čas. Celý testovací proces je omezen hlavně časem. Termín zahájení výroby (ukončení testovacího procesu) je stanoven zákazníkem (např. výrobcem automobilů) a tento termín musí být dodržen. Nemusí se vždy jednat o zahájení výroby, ale může se jednat i o jiné termíny specifikované zákazníkem. Testování nemůže probíhat nekonečně, ale je

nutné najít určitý kompromis mezi časem a hloubkou testování (intenzitou testování), aby testovací proces byl dokončen v určeném termínu a byly dosaženy všechny stanovené cíle.



Obr.č. 2.3 Strategie testování

Na obr.č. 2.3 je znázorněn proces strategie testování. Zákazník zadá požadavky na testování produktu a na základě těchto požadavků jsou určeny priority samotného testování, což je velice důležitá část. Dále zde dochází ke shrnutí požadavků od zákazníka a jejich vyhodnocení. Na základě vyhodnocení požadavků je stanoven druh a požadavky na testovací zařízení. Strategie testování je hlavním řídícím elementem celého testovacího procesu. Výsledkem strategie testování je testovací matice, která je znázorněna v tab. č. 2.1.

V testovací matici můžeme vidět příklad systému venkovního osvětlení. Systém je rozdělen do 3 vlastností a každá tato vlastnost má několik funkcí. Úroveň pokrytí zkoušek se vztahuje k celému systému. Dle těchto úrovní je stanoven přibližný počet testů (viz dále).

Tab. č. 2.1 Testovací matice

System	Vlastnost (Feature)	Funkce (Function)
Venkovní osvětlení		
	Osvětlení	
		tlumená světla
		dálková světla
		nastavení světla
		přední mlhová světla
		automatická světla
	Indikátor	
		zadní mlhová světla
		brzdová světla
		světla při couvání
		zadní světla
	Ukazatel směru jízdy	
		odbočení
		varování
		crash
		zamknutí/odemknutí

Úrovně pokrytí zkoušek (angl. Test coverage levels)

Úrovně pokrytí zkoušek jsou znázorněny v tab. č. 2.2. V tomto případě se jedná o příklad od společnosti MBtech Bohemia. Úrovně se mohou lišit dle požadavku zákazníka. V tabulce je uvedeno 5 úrovní. V praxi je možné použít méně či více úrovní pokrytí zkoušek, opět záleží na požadavcích zákazníka a na zvolené strategii. V průběhu vytváření strategie se určují intenzity testování jednotlivých systémů.

Jak již bylo zmíněno, na tomto příkladu můžeme rozpoznat 5 úrovní pokrytí zkoušek. V případě, že bude vybrán stupeň „N“, testování nebude žádné. Písmenem „S“ je označeno testování s minimální intenzitou, to znamená, že bude proveden jeden test pro jednu vlastnost. Další stupeň je označen písmenem „P“ – částečné pokrytí, kde se provádí jeden test pro jednu funkci. Pro tyto 3 druhy pokrytí není nutné použití metod pro specifikaci testu z důvodu, že počet testů není velký a není nutné počty testů dále minimalizovat. Čtvrtým stupněm ve zmíněném příkladu je písmeno „L“, kde se provádí jeden test na jednu podmínku, zde se doporučuje použití metod pro specifikaci testu. Metodami pro specifikace testů se zabývá

kapitola č. 3. Pátý a poslední stupeň je označen písmenkem „F“, kde je přímo vyžadováno použití metod pro specifikaci testu. Jedná se o úplné pokrytí a je zde nutné počet testů omezit.

Tab. č. 2.2 Úrovně pokrytí zkoušek

Označení	Anglický výraz	Pokrytí	Doporučený počet testů
N	none	žádné	žádný test
S	sparely	minimální	1 test na vlastnost (Feature)
P	partly	částečné	1 test na funkci (Funktion)
L	largely	většinové	1 test na podmínku (Condition)[1]
F	fully	úplné	použití metod: Error Guesing, … [2]

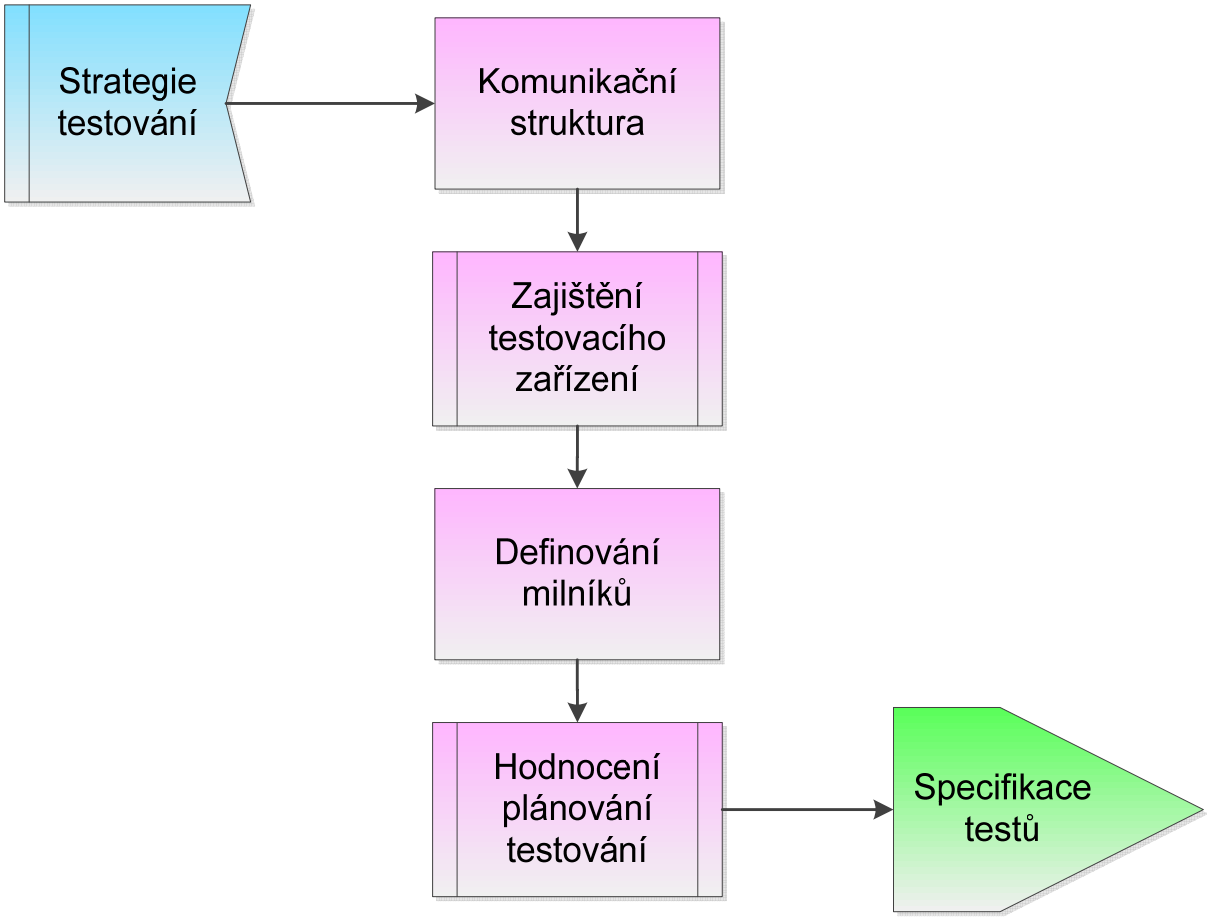
2.4.2 Plánování testovací kampaně

Strategie testování slouží jako podklad pro celý testovací proces a zároveň je také nedílnou součástí plánování. Jak již bylo zmíněno na začátku této kapitoly, všechny části testovacího procesu jsou spolu propojeny a tyto kroky nelze přeskakovat. Na obr.č. 2.4 jsou znázorněny fáze plánování testování. Základem plánování je rozdělení rolí a kompetencí. V této části procesu dojde k vytvoření sohrnů úkolů tzv. „*work packages*“.

Aby bylo možné samotné testování realizovat, je zapotřebí testovací zařízení, které bylo definováno v předešlé části procesu. V této části je nutné toto testovací zařízení zajistit. Dále je v tomto kroku důležité definování rozhraní a komunikační struktury. V průběhu plánování dochází k definování milníků (anglicky - milestone). Milníkem můžeme označit záchytný bod v průběhu procesu. Milníky (mezníky) jsou dávány před konce jednotlivých fází tak, aby v případě výskytu problému bylo možné uskutečnit nápravná opatření a požadavek od zákazníka mohl být splněn v daném termínu. Milníky mohou významným způsobem přispět k plánování a úspěšné realizaci celého testovacího procesu.

[1] Doporučuje se použití metod
[2] Vyžaduje se použití metod

Výstupem plánování testování je test plán. Obsahem tohoto plánu nejčastěji bývá: komunikační plán, seznam testovaných a netestovaných funkcí, hardwarové a softwarové konfigurace, časový plán, odpovědnosti, rizika, apod.



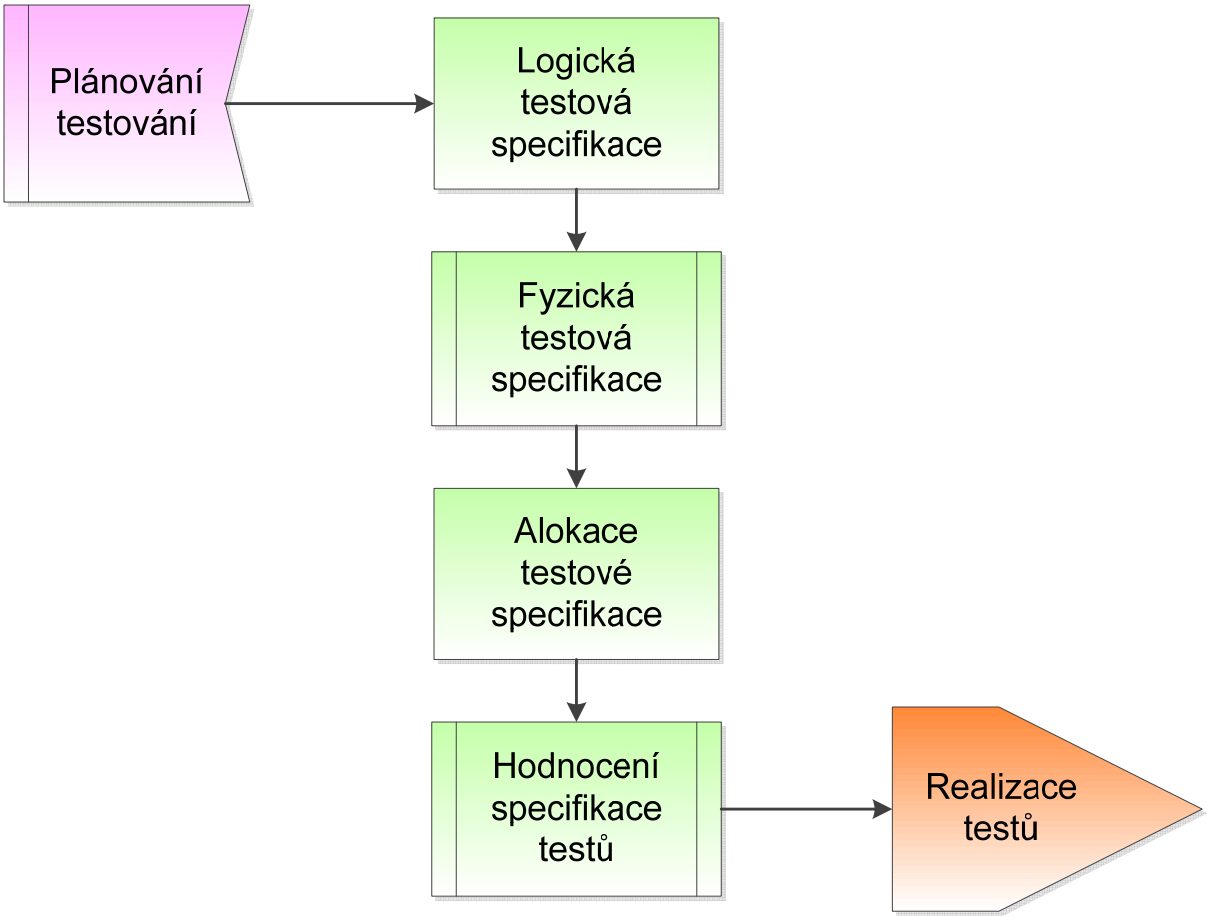
Obr.č. 2.4 Plánování testování

2.4.3 Specifikace testů

Specifikace testu označuje další fázi testovacího procesu. V tomto kroku hraje hlavní úlohu aktivita Test-designéra. Tato část procesu je zobrazena na obr.č. 2.5.

Prvním krokem je vytvoření tzv. logické testové specifikace. Jedná se o samotnou myšlenku tedy popis (slovní nebo písemný) samotné testové specifikace – jakým způsobem se bude testovat.

Následujícím krokem je fyzická testová specifikace, kde je tato myšlenka převedena do určité databáze. Touto databází může být například MS Excel, DOORS, atd., čímž vzniká testová specifikace (angl. - test case).

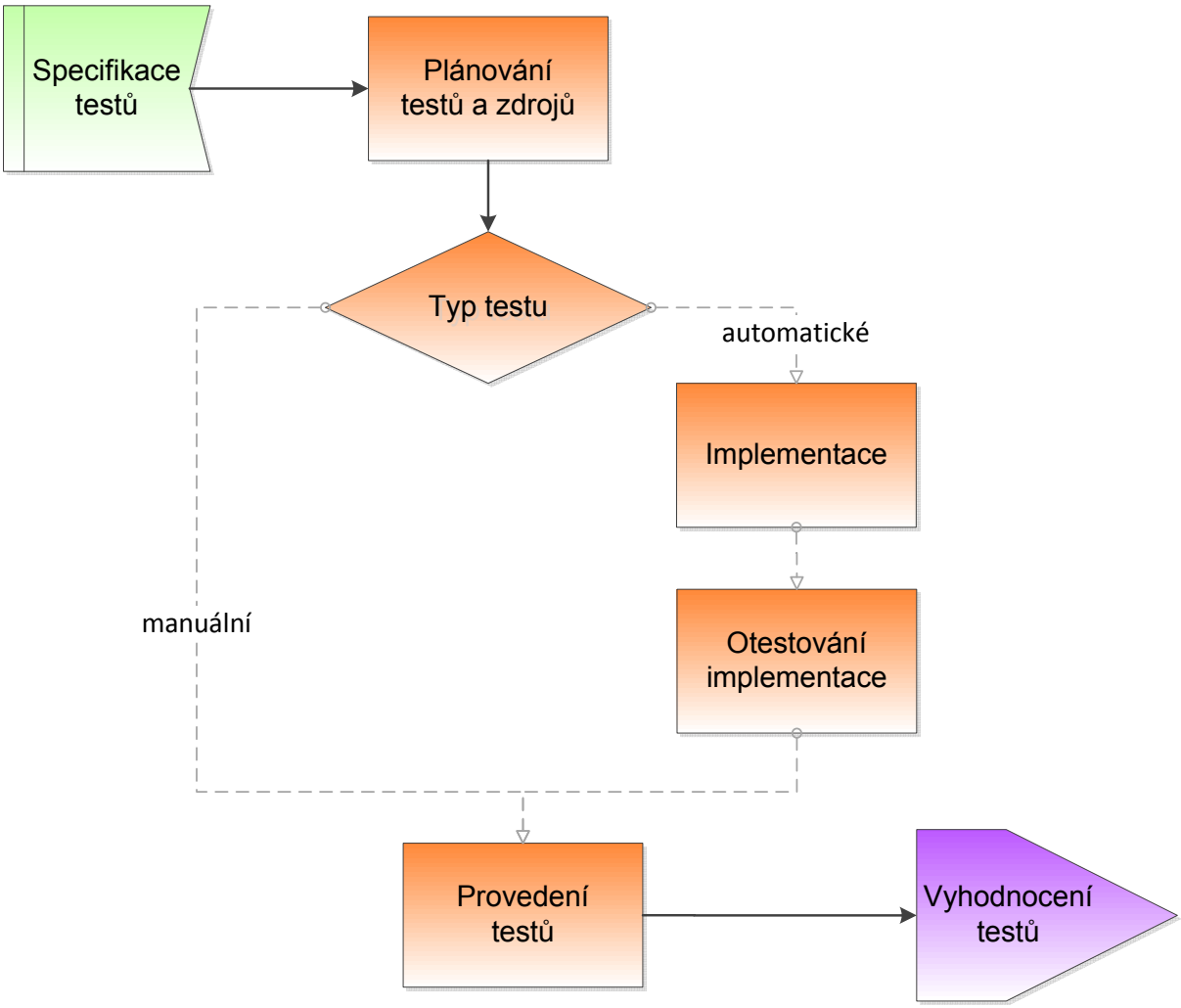


Obr.č. 2.5 Specifikace testu

Alokací se rozumí přiřazení testové specifikace (test case) ke konkrétnímu testovacímu zařízení. Ve strategii testování byla stanovena úroveň pokrytí zkoušek. V případě, že na jejím základě bude vyžadováno použití metod pro specifikaci testů, tyto metody jsou zde aplikovány. Testovou specifikací je stanoven seznam po sobě jdoucích kroků tzn. které činnosti mají být uskutečněny – krok za krokem – jedná se o přesný popis. Specifikace testů je základem pro realizaci testů.

2.4.4 Realizace testů

Realizace testů vychází z testové specifikace a možností testovacího zařízení. Na základě plánování testů a zdrojů se rozhodne, zda se bude jednat o manuální či automatické testování. Na obr.č. 2.6 můžeme zhlédnout postup realizace testů. Po zvolení manuální či automatické cesty je proveden samotný test.



Obr.č. 2.6 Realizace testu

- **Automatické testování**

Výhodou automatického testování je, že při samotném provádění testu je automaticky vytvořen tzv. protokol (report) obsahující kroky testu a dílčí a celkový výsledek. Zároveň je možné test snadno spouštět rekurzivně, neboli opakovaně. S jedním testem lze testovat různé konfigurace. Touto metodou můžeme provádět testy na různých testovacích stavech současně a tím tak snížit dobu testování, čímž se dlouhodobě snižují náklady. Nevýhodou jsou vyšší počáteční investice než u manuálního způsobu testování. U automatického testování jsou testy implementovány do kódu. Rozlišujeme dva druhy kódu:

- Real-time kód

Testování se provádí v reálném čase v testovacím systému a lze provádět více příkazů paralelně.

- PC Code

Tento způsob testování je prováděn na řídícím počítači a nelze spouštět příkazy paralelně.

Není možné automatizovat veškeré testy, ale některé je nutné provést polo-automaticky anebo plně manuálně.

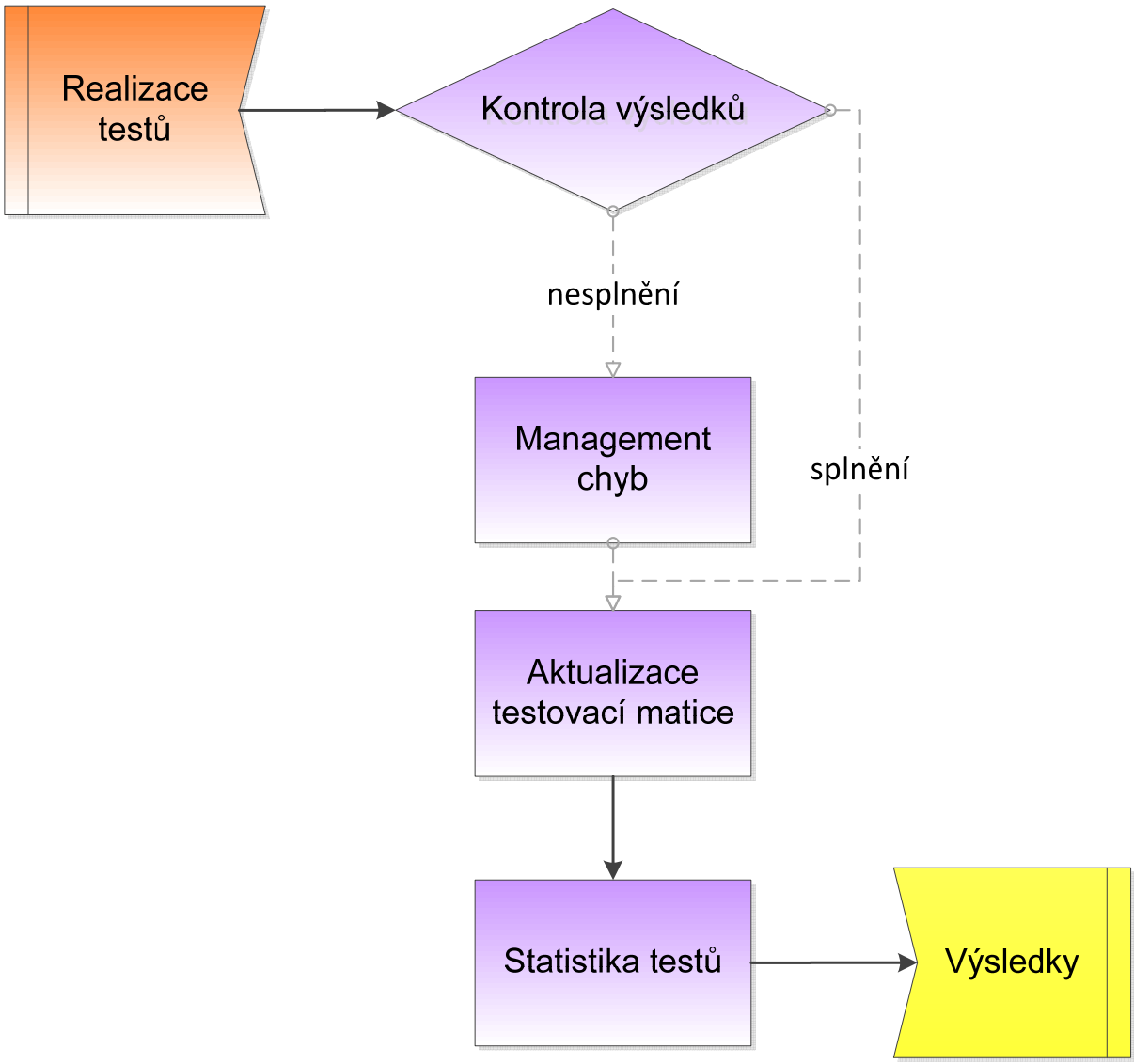
- **Manuální testování**

Nevýhodou manuálního testování je, že všechny protokoly jsou dělány ručně. Jsou zde zapotřebí kontrolní listy. U této metody nelze zaručit stejné (s ohledem na časování) opětovné provádění. Manuální testování může být časově velmi náročné.

2.4.5 Vyhodnocení testů

Posledním úkolem testovacího subjektu je interpretace výsledků provedených testů zpravidla testovacím inženýrem. Po ukončení samotného testování se provádí vyhodnocení jednotlivých výsledků testů. Jednotlivé části posledního kroku testovacího procesu jsou znázorněny na obr.č. 2.7.

V konečné fázi dochází k aktualizaci testovací matice, která byla nadefinována v prvním kroku celého procesu testování – strategii testování. V testovací matici dochází k doplnění výsledků jednotlivých testů. Výsledkem tohoto kroku je test report (statistika všech provedených a zamýšlených testů).



Obr.č. 2.7 Vyhodnocení testu

V tab. č. 2.3 je znázorněn možný příklad aktualizace testovací matice. Zeleně jsou vyznačeny úspěšně provedené testy se správným výsledkem a červeně jsou zabarveny nevyhovující výsledky. Další částí tohoto procesu je zkoumání daného testu, který je označen jako NOK. Musíme vyhodnotit, zda chyba, která se vyskytla, byla způsobena špatnou konfigurací testovacího zařízení nebo případně chybnou specifikací. V těchto dvou případech je nutné zjednat nápravu buď správnou konfigurací systému, nebo úpravou specifikace testu. Po té musí být test proveden ještě jednou. V případě, že se jednalo o skutečnou chybu testovaného systému (řídící jednotky, atd.) je nutné tuto chybu zanést do managementu chyb.

Tab. č. 2.3 Aktualizovaná testovací matice

Funkce (Function)	Název testu	Výsledek
Tlumená světla		
	Tlumená světla 1	OK
	Tlumená světla 2	NOK
	Tlumená světla 3	OK
Dálková světla		
	Dálková světla 1	OK
	Dálková světla 2	Zatím neproveden
Nastavení světla		
	Nastavení světla 1	OK
	Nastavení světla 2	OK
Přední mlhová světla		
	Přední mlhová světla 1	NOK
	Přední mlhová světla 2	OK
Automatická světla		
	Automatická světla 1	Zatím neproveden
	Automatická světla 2	OK

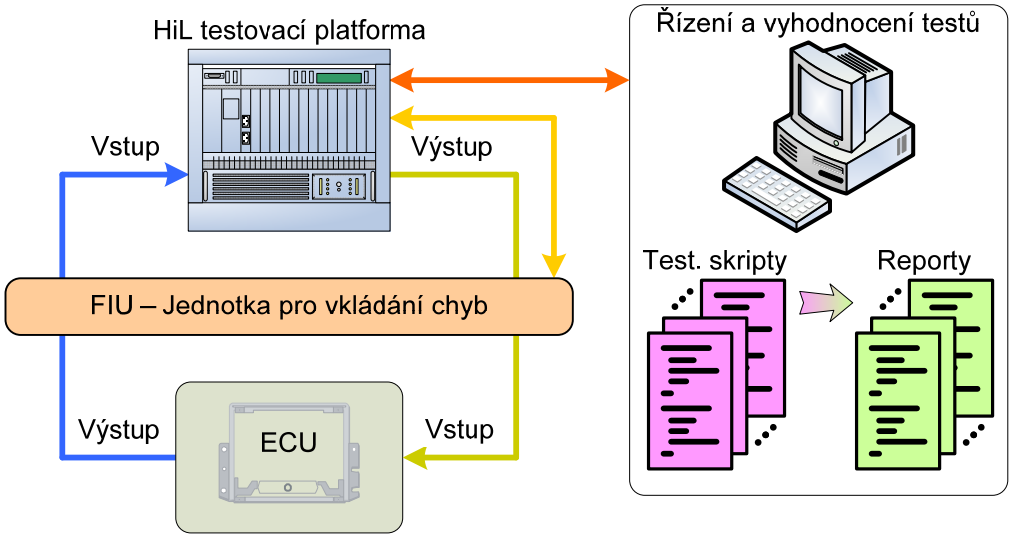
2.5 HiL testování

Předmětem testování je především samotná jednotka, u které je nezbytné mít možnost libovolně nastavovat parametry okolí jednotky. Toho můžeme dosáhnout pomocí speciálního konfigurovatelného okolí nebo simulace tohoto okolí. Simulace umožňuje mnohem širší rozsah parametrizace, snadno generovat i nereálné stavy či hodnoty a bývá ve většině případů jednodušší na realizaci. Můžeme říci, že testovaná řídící jednotka je během testování zapojena ve smyčce, buď k modifikovanému reálnému či simulovanému okolí. Poté o takovém testování mluvíme jako o testování hardwaru ve smyčce – HiL testování (z anglického Hardware in the Loop) [4].

HiL testování je jedním z kroků procesu rychlého vývoje. Tento proces je často využíván výrobci elektronických řídících jednotek (ECU), aby se zkrátila doba vývoje a snížily náklady. Zavedením HiL testování můžou výrobci získávat informace z celého procesu vývoje dané řídící jednotky [4].

2.5.1 Princip HiL testování

Na obr. č. 2.8 je znázorněn základní princip HiL testování. Je zde zobrazen testovaný systém – elektronická řídící jednotka (ECU), která má různé vstupy a výstupy a je připojena ve smyčce k testovací platformě ve spojení s počítačem typu PC. Mezi řídící jednotku a testovací platformu může být někdy zařazena jednotka pro vkládání chyb (FIU – failure insertion unit), která ve většině případů pomocí relé může umožnit simulovat poruchy typu rozpojení a zkrat na záporný nebo kladný pól napájení, což vyžaduje nejméně jeden rozpínací a dva spínací nezávislé kontakty na jeden ovlivňovaný vstup / výstup. U simulace chyby typu zkrat mezi vybranými vodiči se využívá dalších kontaktů připojující vybrané vodiče na sběrnicové vedení. Tím může být docíleno zkratu mezi více vodiči najednou, avšak na druhou stranu je počet sběrnicových vodičů omezen, jelikož by se jinak neúnosně zvyšoval počet potřebných spínacích kontaktů (relé), které jednotka FIU musí obsahovat [4].



Obr. č. 2.8 Princip HiL testování[4]

Samotný princip testování je přibližně následující. Testovací systém stimuluje prostřednictvím podnětů vstupy řídící jednotky a na výstupech zachytává odezvy. Jsou-li předmětem testování zasíťované řídící jednotky, které spolu komunikují prostřednictvím nějaké datové sítě (např. sběrnice CAN, apod.), tak testovací systém provádí sledování této komunikace na sběrnici. Pro zachování časové souslednosti je vhodné, aby testovací platforma přiradila ke všem událostem (vydaný podnět,

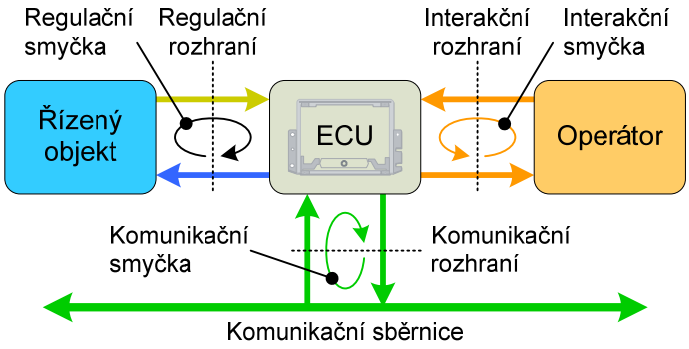
zachycená odezva nebo přijatá zpráva ze sběrnice) určitou časovou značku. Tyto značky usnadňují vyhodnocování testování a měření např. doby odezvy [4].

Pro časování v reálném čase je zapotřebí, aby testovací systém byl schopen provozu jako tzv. real-time hardware. Počítač typu PC s operačním systémem Microsoft Windows je pro tento záměr nepoužitelný z toho důvodu, že je nemožné zajistit přesnější časování jednotlivých operací, jelikož nedokáže zajistit časování 1ms a kratší, které je pro testování požadováno v praxi. Z tohoto důvodu je v rámci testovacího systému mezi počítač PC a testovaný systém zařazena HiL testovací platforma, která tvoří rozhraní pro přímé napojení na testovaný systém – disponuje potřebnými vstupy/výstupy a komunikačními rozhraními, dále pak zajišťuje požadované časování všech událostí [4].

Řízení testování může v některých případech zahrnovat i určité simulační výpočty, které jsou nutné pro zajištění správné funkce testovaných jednotek. Tyto výpočty bývají celkem složité a náročné na výpočetní výkon testovacího systému. Z tohoto důvodu je ve většině případů výhodné přesunout simulační výpočty přímo do testovací platformy a tím tak zlepšit rozložení výkonu testovacího systému. Testovací platformu je nutné navrhnout jako programovatelný testovací a simulační systém s vlastním výpočetním výkonem. Takový systém se jmenuje HiL simulátor [4].

2.5.2 Implementace HiL testování

HiL testovací platforma může být implementována např. jako jednoduché rozhraní mezi testovanou řídící jednotkou a počítačem PC, které zajišťuje generování podnětů, zachytávání odezev, sledování případné komunikace a časování všech těchto operací. Vyhodnocování a řízení testování je prováděno počítačem PC, viz obr. č. 2.8 [4].



Obr. č. 2.9 Komplexní řídicí systém a jeho rozhraní[4]

Je-li předmětem testování řídící jednotka, která představuje komplexní řídicí systém, např. jaký je zobrazen na obr. č. 2.9, pak je nezbytné, aby testovací systém byl schopen uzavřít všechny smyčky pro zajištění normální funkce řídící jednotky. Uzavření smyček lze provést např. modelem okolí řídící jednotky a to většinou pro každou smyčku odděleně. Testovací systém pak simuluje model okolí v reálném čase. V případě použití počítače PC s příslušným rozhraním, pak musí tyto simulační výpočty provádět tento počítač. Vzhledem k náročnosti simulačních výpočtů na výpočetní výkon je v některých případech výhodné přemístit některé simulační výpočty (zejména ty týkající se regulační smyčky) do HiL testovací platformy. To lze realizovat v případě, kdy HiL testovací platforma je implementována tak, že disponuje vlastním výpočetním výkonem, což je většina dodávaných profesionálních systémů pro testování [4].

Při rozhodování, jakou implementaci testovacího systému zvolit – zda zakoupit profesionální tester nebo vyvinout vlastní rozhraní k počítači PC, jsou kromě technických parametrů rozhodující především ekonomické a časové aspekty. Profesionální testery jsou univerzální, nechají se dobře naprogramovat, modely se (většinou) implementují pomocí robustního, ale komerčního, nástroje Matlab/Simulink, avšak pořizovací náklady jsou značné – v řádech milionů Kč. Ovšem implementace vlastního testovacího rozhraní je naproti tomu výrazně levnější. Ale pokud je doba vývoje příliš dlouhá, mohou náklady dosáhnout úrovně nákladů profesionálního zařízení. Navíc doba implementace je prodloužena právě o dobu vývoje[4].

3 Metody pro specifikaci testů

Metody pro specifikaci testu snižují ve většině případů náklady ale i pracnost. Cílem je obvykle snížit počet jednotlivých testů, protože testování nemůže trvat nekonečně dlouho. Techniky specifikace mohou dát odpovědi na otázky, jaké testovací případy by měly být vytvořeny, jaké testovací případy budou vytvořeny, proč byly vytvořeny právě tyto testovací případy a hlavně kolik času je zapotřebí k vytvoření testovacího případu. V tab. č. 3.1 jsou k nahlédnutí metody pro specifikace testu, které jsou popsány v následujících kapitolách.

Tab. č. 3.1 Metody pro specifikaci testů

Metody pro specifikaci	Typické použití
MCDC	Komplexní rozhodnutí s vysokou prioritou
Třídy ekvivalence	Funkce s analogovým vstupním parametrem
Strukturní testy	Kontrolní tokové struktury: white-box a black box
Analýza hraničních hodnot	Podrobné vyhodnocování kontiuálních parametrů
"Error Guessing"	Chyba s vysokou prioritou
Klasifikační strom	Základ pro další metody: MCDC, Analýza hraničních hodnot
"Gutfall"	Testy s nízkou prioritou, regresní test při změně jiných modulů

Testovací případ (ang. Test case)

V průběhu celého testovacího procesu jsou odhalovány chyby. Největší možností, jak najít a odhalit chybu, je využít cílené testování založené na zdokumentovaných testovacích případech. Testovací případ, často se využívá i anglický výraz „test case“, se vytváří jak pro manuální tak i pro automatizované testy. Testovací případy jsou vždy tvořeny seznamem prováděných kroků a očekávaných výsledků. Pokud je testovací případ vybrán pro automatizaci je nutné ho následně implementovat. Takto implementovaný testovací případ se často nazývá testovací skript. Ten je tvořen sadou programových instrukcí a na rozdíl od manuálního testu by měl být schopen sám rozpoznat, zda uspěl či selhal. Testovací případ je dokument, popisující určitou činnost, kterou je nutno otestovat. Obecně lze říci, že obsahuje kroky se skutečnými vstupními hodnotami spolu s očekávanými výstupy [5].

White-Box testování

Pro úplné otestování produktu bychom potřebovali pomocí testovacích případů otestovat všechny možné logické cesty. I po otestování všech logických cest se mohou vyskytnout defekty, protože některé logické cesty mohou chybět a nemusejí být nalezeny defekty citlivé na data [5].

Black-Box testování

Tester na produkt (vozidlo) pohlíží jako na černou skříňku s danou specifikací, vnitřní struktura a funkce produktu ho nezajímají. Hledá případy, ve kterých se produkt nechová podle specifikace. Pro nalezení všech defektů by bylo nutné otestovat vozidlo se všemi možnými kombinacemi vstupů (platnými i neplatnými), což je prakticky nemožné [5].

Při White-Box i Black-Box testování se budou testovací případy skládat z popisu vstupních dat a z popisu správného výstupu pro daná vstupní data. Testovací případy by měly být uloženy z toho důvodu, že je můžeme opakovaně využívat (např. pro otestování produktu po změně) [5].

3.1 Formální techniky

Pro tyto metody existují přísná pravidla pro odvození testovacích případů. V tomto případě mohou být učiněna jasná prohlášení o pokrytí testů. Odvození testovacích případů je nezávislé na testovacím inženýrovi. Jsou zde kladeny vyšší nároky na bázi podkladů pro testování. U metod formálních technik je tvorba testovacích případů často náročnější na pracovní nasazení.

3.1.1 Třídy ekvivalence

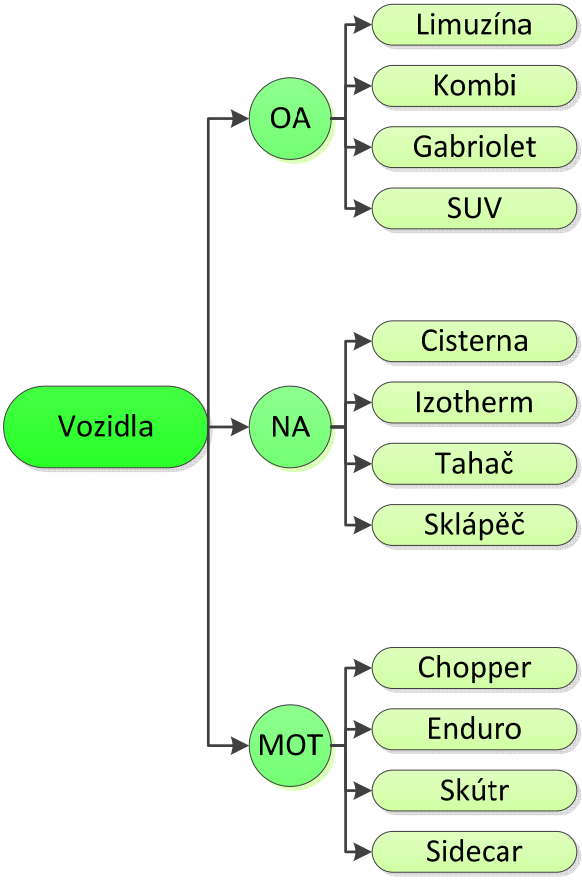
Klasifikace je zobrazení třídy (třídy ekvivalence) nebo kategorie, která je sestavena na základě určitých principů pořadí. Třídy ekvivalence nebo kategorie označují soubor objektů, které jsou seskupeny na základě společných vlastností / parametrů do skupin. Klasifikace vytváří „pořádek“ v komplexních situacích. Třídy ekvivalence slouží jako podklad pro další metody (např. MCDC).

Úplné otestování vozidla není možné, proto je pro testování velmi podstatný návrh efektivních testovacích případů. Klademe si otázku - jaká podmnožina všech teoreticky možných testovacích případů má největší pravděpodobnost nalézt většinu defektů? První možností je náhodně vybraná podmnožina všech možných vstupů. Jedná se nejspíše o jednu z nejhorších možností, protože má malou pravděpodobnost být optimální podmnožinou nebo být alespoň blízká optimální podmnožině. Druhým případem je výběr testovacích případů pomocí metod. Existuje několik metodik, každá má své silné a slabé stránky - tj. každá detekuje / přehlédne jiné typy defektů proto je dobré připravovat testovací případy aplikováním více metod.

Dobrý testovací případ bude vyvolávat co nejvíce vstupních podmínek a tím omezí celkový počet potřebných testovacích případů. Testovací případ by měl pokrývat určitou množinu vstupních hodnot. Množinu vstupů bychom měli rozdělit do tříd ekvivalence tak, abychom mohli rozumně předpokládat, že test nějaké reprezentativní hodnoty dané třídy je ekvivalentní testu kterékoli další hodnoty.

Vezmeme každou vstupní podmínku a podle ní rozdělíme množinu všech vstupních hodnot do dvou nebo více podmnožin. Existují dva typy tříd ekvivalence - platné (reprezentující platné vstupy) a neplatné (reprezentující chybné vstupní hodnoty). Příklad tříd ekvivalence je vyznačen na obr.č. 3.1. Rozdělení do tříd ekvivalence je heuristický proces, můžeme využít následujících doporučení:

- Pokud vstupní podmínka specifikuje interval hodnot (např. rok může být mezi 1900 a 2000), pak máme jednu platnou třídu ekvivalence (hodnoty 1900 až 2000) a dvě neplatné třídy ekvivalence (hodnoty < 1900, hodnoty > 2000).



Obr.č. 3.1 Třídy ekvivalence

- Pokud vstupní podmínka specifikuje množinu vstupních hodnot a pokud lze předpokládat, že každá z nich bude obsluhována jinak (např. "OA - osobní automobil", "NA - nákladní automobil"), bude jedna platná třída ekvivalence pro každý prvek množiny; přidáme jednu neplatnou třídu ekvivalence pro další prvek množiny (např. "MOT - motocykl"). Toto doporučení je znázorněno na obr.č. 3.1.

- Pokud vstupní podmínka specifikuje situaci, která "musí nastat", např. první znak identifikátoru musí být písmeno, bude jedna platná třída ekvivalence (je písmeno) a jedna neplatná třída ekvivalence (není písmeno).

Pokud je důvod předpokládat, že prvky nějaké třídy nejsou obsluhovány stejně, rozdělíme třídu do menších tříd ekvivalence.

Příklad - „potkávací světlo“

Pro znázornění třídy ekvivalence jsem vybrala příklad na potkávací světlo. Vycházíme z níže uvedené rovnice, dle které je doplňována tab. č. 3.2. Každý prvek je možné použít pouze jednou a tím se eliminuje počet testů. Po vyčerpání všech možností může skončit vytváření testovacích případů. Zelenou fajfkou jsou označeny prvky, které byly v tomto kroku vybrány. Po dosazení do rovnice se stanoví očekávaný výsledek testu (např. potkávací světla – zapnuto). V každém kroku je použita tato rovnice:

$$(V = Zap\ nebo (V = Auto\ a (SSenz = Tma\ nebo\ DSenz = Mokro)))a(Z > Zap.) \rightarrow PS$$

1. krok(tab. č. 3.2):

Vypínač světel (V): vypnuto, automaticky, zapnuto

Světelný senzor (SSenz): světlo, tma

Dešťový senzor (DSenz): mokro, sucho

Zapalování (Z): vypnuto, zapnuto, start

Tab. č. 3.2 Potkávací světlo – 1.krok

Testovací případ	Vypínač	Světelný senzor	Dešťový senzor	Zapalování	Potkávací světla
TP 1	Zapnuto	Světlo	Sucho	Zapnuto	Zapnuto

2. krok (tab. č. 3.3):

Vypínač světel (V): vypnuto, automaticky, zapnuto

Světelný senzor (SSenz): světlo, tma

Dešťový senzor (DSenz): mokro, sucho

Zapalování (Z): vypnuto, zapnuto, start

Tab. č. 3.3 Potkávací světlo – 2.krok

Testovací případ	Vypínač	Světelný senzor	Dešťový senzor	Zapalování	Potkávací světla
TP 1	Zapnuto	Světlo	Sucho	Zapnuto	Zapnuto
TP 2	Vypnuto	Tma	Mokro	Start	Vypnuto

3. krok (tab. č. 3.4):

Vypínač světel (V): vypnuto, automaticky, zapnuto

Světelný senzor (SSenz): světlo, tma

Dešťový senzor (DSenz): mokro, sucho

Zapalování (Z): vypnuto, zapnuto, start

Tab. č. 3.4 Potkávací světlo – 3.krok

Testovací případ	Vypínač	Světelný senzor	Dešťový senzor	Zapalování	Potkávací světla
TP 1	Zapnuto	Světlo	Sucho	Zapnuto	Zapnuto
TP 2	Vypnuto	Tma	Mokro	Start	Vypnuto
TP 3	Auto	Tma	Sucho	Zapnuto	Zapnuto

4. krok (tab. č. 3.5):

Vypínač světel (V): vypnuto, automaticky, zapnuto

Světelný senzor (SSenz): světlo, tma

Dešťový senzor (DSenz): mokro, sucho

Zapalování (Z): vypnuto, zapnuto, start

Tab. č. 3.5 Potkávací světlo – 4.krok

Testovací případ	Vypínač	Světelný senzor	Dešťový senzor	Zapalování	Potkávací světla
TP 1	Zapnuto	Světlo	Sucho	Zapnuto	Zapnuto
TP 2	Vypnuto	Tma	Mokro	Start	Vypnuto
TP 3	Auto	Tma	Sucho	Zapnuto	Zapnuto
TP 4	Auto	Světlo	Mokro	Zapnuto	Zapnuto

5. krok (tab. č. 3.6):

Vypínač světel (V): vypnuto, automaticky, zapnuto

Světelný senzor (SSenz): světlo, tma

Dešťový senzor (DSenz): mokro, sucho

Zapalování (Z): vypnuto, zapnuto, start

Tab. č. 3.6 Potkávací světlo - 5.krok

Testovací případ	Vypínač	Světelný senzor	Dešťový senzor	Zapalování	Potkávací světla
TP 1	Zapnuto	Světlo	Sucho	Zapnuto	Zapnuto
TP 2	Vypnuto	Tma	Mokro	Start	Vypnuto
TP 3	Auto	Tma	Sucho	Zapnuto	Zapnuto
TP 4	Auto	Světlo	Mokro	Zapnuto	Zapnuto
TP 5	Zapnuto	Světlo	Mokro	Vypnuto	Vypnuto

Výstupem použití metody ekvivalentních tříd na zvoleném příkladu je specifikace 5 testů. Tyto testy jsou zobrazeny v tab. č. 3.6. Pro kontrolu, zda byly vybrány všechny možnosti, je zde uveden řádek Suma, kde jsou sepsány všechny prvky daného systému (např. Vypínač – zapnuto, vypnuto, auto).

$$(V = Zap\ nebo(V = Auto\ a(SSenz = Tma\ nebo\ DSenz = Mokro)))a(Z > Zap.) \rightarrow PS$$

6. krok (tab. č. 3.7):

Vypínač světel (V): vypnuto, automaticky, zapnuto

Světelný senzor (SSenz): světlo, tma

Dešťový senzor (DSenz): mokro, sucho

Zapalování (Z): vypnuto, zapnuto, start

Tab. č. 3.7 Potkávací světlo - 6.krok

Testovací případ	Vypínač	Světelný senzor	Dešťový senzor	Zapalování	Potkávací světla
TP 1	Zapnuto	Světlo	Sucho	Zapnuto	Zapnuto
TP 2	Vypnuto	Tma	Mokro	Start	Vypnuto
TP 3	Auto	Tma	Sucho	Zapnuto	Zapnuto
TP 4	Auto	Světlo	Mokro	Zapnuto	Zapnuto
TP 5	Zapnuto	Světlo	Mokro	Vypnuto	Vypnuto
Suma	Zapnuto, Vypnuto, Auto	Světlo, Tma	Mokro, Sucho	Vypnuto, Zapnuto, Start	Vypnuto, Zapnuto

3.1.2 MCDC

Další velice používanou metodou je MCDC (angl. Modified Condition/Decision Coverage) neboli v překladu modifikované pokrytí podmínek a/nebo rozhodnutí, která se uplatňuje nejen při testování v automobilovém průmyslu, ale i při testování kritických systémů v leteckém průmyslu. Princip spočívá ve snížení počtu testovacích případů z 2^n, kde n je počet podmínek (Booleovských proměnných nebo výrazů porovnání), tedy z triviálního testu, na počet n + 1 se zachováním pokrytí požadovaném i v kritických aplikacích. Metoda je velmi dobře popsána v [6].

Metoda MCDC pracuje se symboly, které jsou uvedené v tab. č. 3.8.

Tab. č. 3.8 Používané symboly

Anglické označení	České označení	Symbol
AND	A	∧
OR	NEBO	∨
Boolean variable	Booleovské proměnné	**A**

Znázornění (Condition = podmínka)

Booleovský výraz neobsahuje žádný logický operátor.

A = {0; 1} - je podmínka

A and B = {0; 1} - není podmínka

Druhý výraz není podmínkou z toho důvodu, že je zde používán logický operátor.

Znázornění (Decision = rozhodnutí)

Booleovský výraz se skládá z podmínek a z jednoho nebo více logických operátorů. Podmínky, které jsou používány vícekrát v průběhu rozhodování, budou pokaždé rozpoznány.

(A and B) or C - je rozhodnutí

V tomto případě rozhodnutí zahrnuje dva logické operátory (and, or) a tři podmínky A, B a C.

(A and B) or (A and C) - je rovněž rozhodnutí

V druhém případě rozhodnutí obsahuje 3 logické operátory (and, or, and) a čtyři podmínky A, B, A, a C.

MCDC - podmínky

MCDC představuje praktický přístup pro posouzení logických konstrukcí, zda daný soubor požadavků na bázi testovacích případů odpovídá třem podmínkám, které musí být splněny:

1.) Každá podmínka musí zahrnovat všechny platné hodnoty.
2.) Každé rozhodnutí musí zahrnovat všechny platné hodnoty.
3.) Každá podmínka v rozhodnutí musí být použita tak, aby v případě změny rozhodnutí byl změněn i výsledek (výsledná hodnota).

Redukce testovacího úsilí

Po aplikaci metody MCDC je počet testovaných funkcí zřetelně nižší než při testování všech možných kombinací. Podstatou této metody je výrazná redukce testovacích případů, ale zároveň zachování určité bezpečnosti. Úplné otestování všech kombinací je vyjádřeno $\mathbf{2^n}$ testovacích případů. Po aplikování modifikovaného pokrytí podmínek a rozhodnutí je výsledné snížení testů velice znatelné a je formulováno jako **n + 1** při předpokládaném riziku.

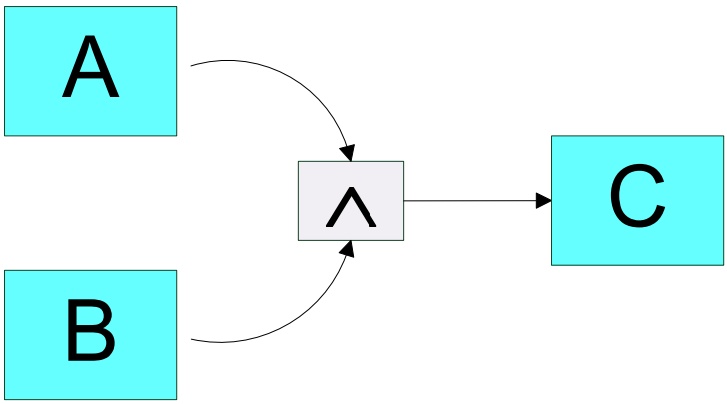
Příklad - redukce testů

Počet podmínek: n = 10

Veškeré kombinace: 2^{10} = 1024 testovacích případů

MCDC: 10 + 1 = 11 testovacích případů

Úvodní příklad k metodě MCDC



$$A\ and\ B = C$$

Obr.č. 3.2 Úvodní příklad k metodě MCDC

Na obr.č. 3.2 je znázorněn základní příklad k metodě MCDC. Jedná se o 2 podmínky, 1 rozhodnutí a jeden logický operátor (A a B = C). Z toho vyplývá, že n = 2 a počet testovacích případů bez použití metody MCDC je $(2^n) = 2^2 = 4$. Tento model si nejprve převedeme do pravdivostní tabulky (tab. č. 3.9) a dále postupuje dle níže uvedených pravidel.

Tab. č. 3.9 Pravdivostní tabulka

A	B	C
AND		
0	0	0
0	1	0
1	0	0
1	0	0

Pravidla:

1.) Všechny podmínky obsahují všechny možné hodnoty.

2.) Všechny rozhodnutí mají všechny možné hodnoty.

3.) Každá podmínka je relevantní alespoň pro jedno rozhodnutí.

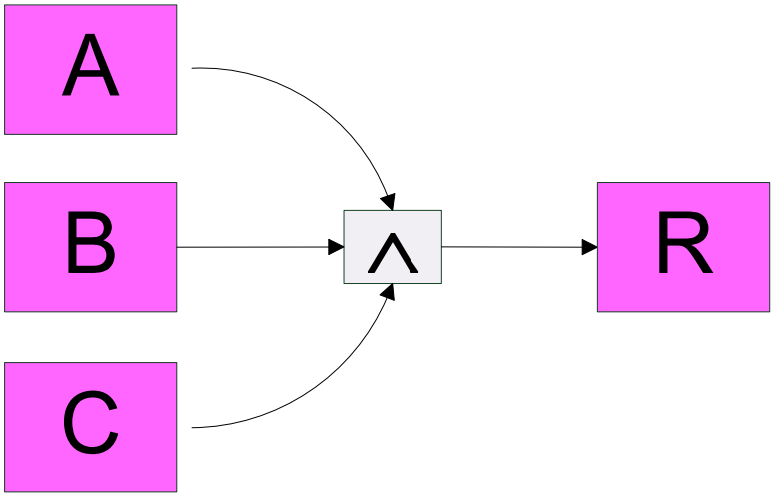
Tab. č. 3.10 Výsledná tabulka – úvodní příklad

Testovací případ	A	B	C
	AND		
~~TP1~~	~~0~~	~~0~~	~~0~~
TP2	0	1	0
TP3	1	0	0
TP4	1	1	1

Ve výsledné tabulce (tab. č. 3.10) lze vyškrtnout testovací případ číslo 1 (TP1), protože tento typ testu není nutné testovat (obsahuje samé nuly). Pokud je to možné, tak při výběru dáváme přednost 1, tím je také možno dosáhnout nižšího počtu testů.

Jak již bylo zmíněno, původní počet testovacích případů je $(2^n) = 2^2 = 4$. Po aplikování MCDC metody se sníží počet testů na (n + 1) = 2 + 1 = 3.

Příklad – AND – rozhodnutí



$$A\ and\ B\ and\ C = R$$

Obr.č. 3.3 Příklad – AND - rozhodnutí

V tomto příkladu, který je zobrazen na obr.č. 3.3, jsou použity 3 podmínky (n = 3), 1 rozhodnutí a 1 logický operátor (A a B a C = R). Počet možných kombinací testovacích případů je $(2^n) = 2^3 = 8$.

Pravidla:

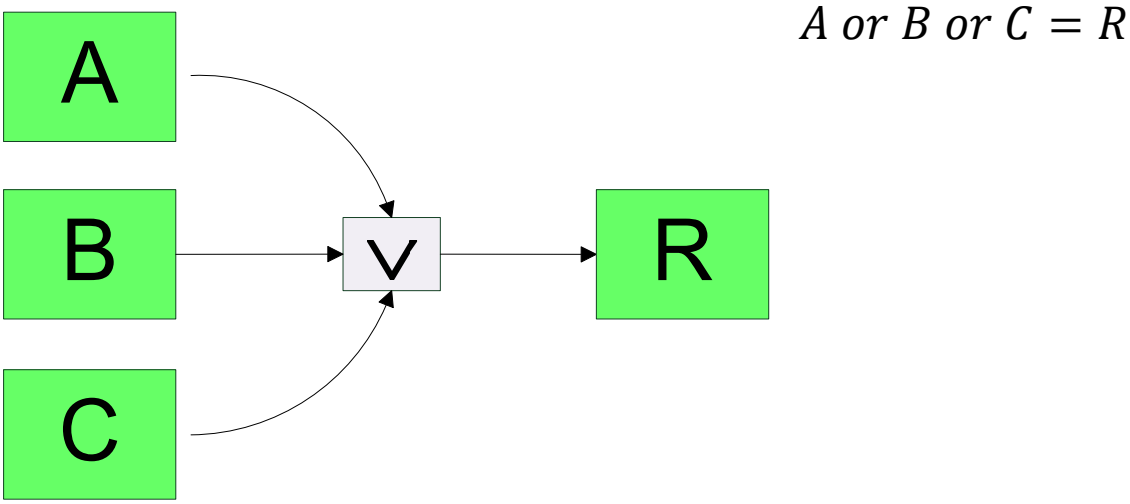
- Podmínka musí být pro rozhodnutí nezbytná.
- Každá podmínka musí zahrnovat všechny možné hodnoty.
- Každé rozhodnutí musí obsahovat všechny možné hodnoty.

Po aplikování metody MCDC dojde k redukci testovacích případů z 2^n na n + 1 a v tomto případě bude konečný počet testů 4. Po zavedení pravidel této metody škrtneme testovací případy TP1, TP2, TP3 a TP5, které jsou uvedeny v tab. č. 3.11. Pokud bychom změnili jednu podmínku (A, B, C), ovlivníme tím i konečný výsledek.

Tab. č. 3.11 Příklad – AND - rozhodnutí

Testovací případ	A	B	C	R
	AND			
~~TP1~~	~~0~~	~~0~~	~~0~~	~~0~~
~~TP2~~	~~0~~	~~0~~	~~1~~	~~0~~
~~TP3~~	~~0~~	~~1~~	~~0~~	~~0~~
TP4	0	1	1	0
~~TP5~~	~~1~~	~~0~~	~~0~~	~~0~~
TP6	1	0	1	0
TP7	1	1	0	0
TP8	1	1	1	1

Příklad – OR – rozhodnutí



Obr.č. 3.4 Příklad – OR - rozhodnutí

Na obr.č. 3.4 je zobrazen další možný příklad, ve kterém jsou použity 3 podmínky (n = 3), 1 rozhodnutí a 1 logický operátor (A nebo B nebo C = R). Počet možných kombinací testovacích případů je stejný jako u příkladu OR - $(2^n) = 2^3 = 8$.

Pravidla:

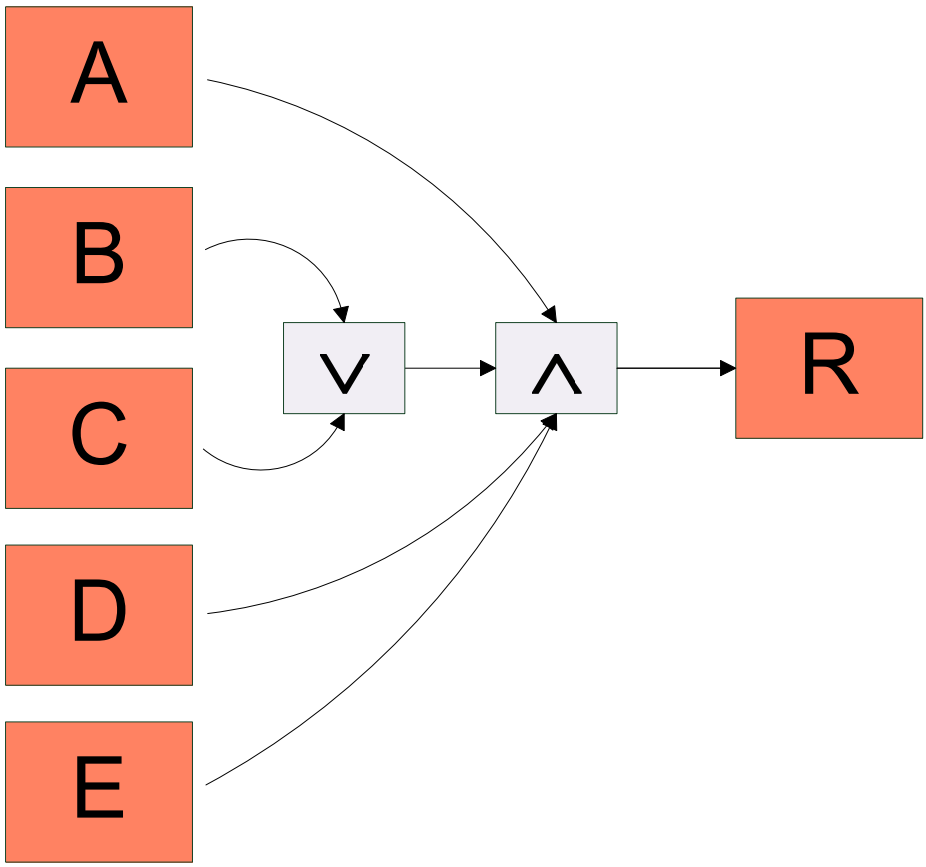
- Podmínka musí být pro rozhodnutí nezbytná.
- Každá podmínka musí zahrnovat všechny možné hodnoty.
- Každé rozhodnutí musí obsahovat všechny možné hodnoty.

Po zavedení pravidel této metody škrtneme testovací případy TP4, TP6, TP7 a TP8, které jsou uvedeny v tab. č. 3.12. Konečný počet testovacích případů je n + 1 = 4, stejně jako v předchozím případě. Pokud bychom změnili jednu podmínku (A, B, C), ovlivníme tím i konečný výsledek.

Tab. č. 3.12 Příklad – OR - rozhodnutí

Testovací případ	A	B	C	R
	OR			
TP1	0	0	0	0
TP2	0	0	1	1
TP3	0	1	0	1
~~**TP4**~~	~~0~~	~~1~~	~~1~~	~~1~~
TP5	1	0	0	1
~~**TP6**~~	~~1~~	~~0~~	~~1~~	~~1~~
~~**TP7**~~	~~1~~	~~1~~	~~0~~	~~1~~
~~**TP8**~~	~~1~~	~~1~~	~~1~~	~~1~~

Složitější příklad – kombinace AND a OR – rozhodnutí



Obr.č. 3.5 Složitější příklad – kombinace AND a OR - rozhodnutí

$$A\ and\ (B\ or\ C) and\ D\ and\ E = R$$

Na obr.č. 3.5 je znázorněna kombinace dvou logických operátorů – AND a OR (a a nebo). V tomto příkladu lze rozpoznat 5 podmínek (A - E) n = 5. Počet testovacích případů kompletních kombinací všech hodnot je $2^5 = 32$. Pomocí metody MCDC je možné snížit počet testů a zároveň zachovat určitou bezpečnost. Abychom mohli testy zredukovat je nutné dodržovat pravidla této metody.

Tab. č. 3.13 Pravdivostní tabulka – rozhodovací úrovně

R (rozhodnutí)									
Správně					**Chybně**				
A	**B**	**C**	**D**	**E**	**A**	**B**	**C**	**D**	**E**
1					**0**				
	1					**0**			
		1					**0**		
			1					**0**	
				1					**0**

Nejprve si připravíme tab. č. 3.13, do které vyplníme diagonály, při správném výsledku jedničky a při chybném výsledku nuly. Hodnotám v diagonále říkáme rozhodovací úrovně. Pokud bychom změnili hodnotu na diagonále v opačnou, výsledek by byl také ovlivněn. Dalším krokem je doplnění 1.části tabulky (Správně) vzhledem k rozhodovací úrovni a hodnoty do 2.části (Chybně) lze opsat. Vyplněné hodnoty jsou uvedeny v tab. č. 3.14.

Tab. č. 3.14 Doplněná pravdivostní tabulka

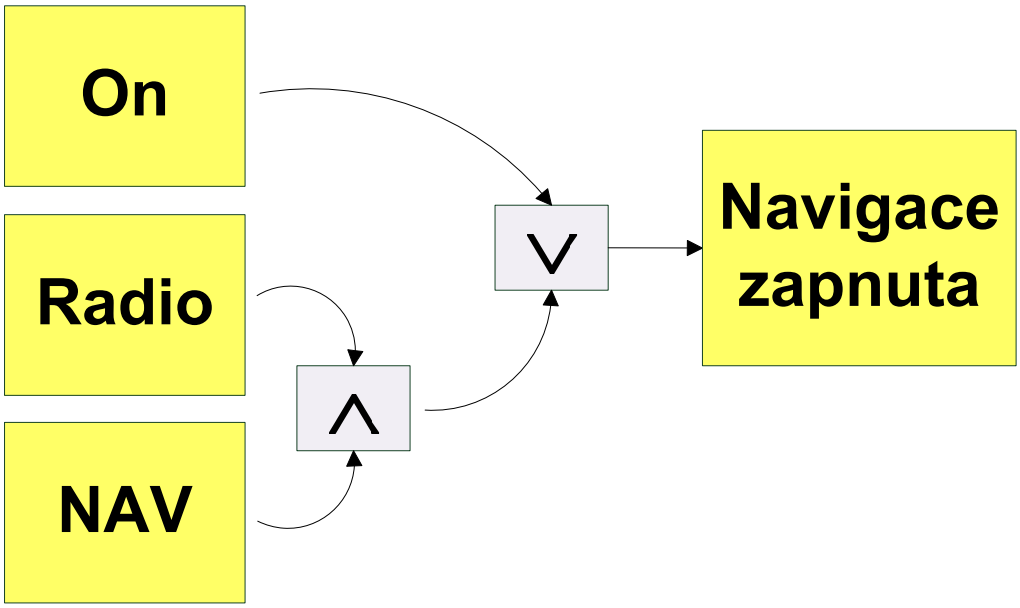
R (rozhodnutí)									
Správně					**Chybně**				
A	**B**	**C**	**D**	**E**	**A**	**B**	**C**	**D**	**E**
1	0	1	1	1	**0**	0	1	1	1
1	**1**	0	1	1	1	**0**	0	1	1
1	0	**1**	1	1	1	0	**0**	1	1
1	0	1	**1**	1	1	0	1	**0**	1
1	0	1	1	**1**	1	0	1	1	**0**

Z tab. č. 3.15 je patrné, že počet testovacích případů je 10 (> n + 1) a to neodpovídá výslednému efektu metody MCDC. Pokud se některé řádky shodují, můžeme je vyškrtnout a tím dosáhnout konečného optimálního výsledku – počet testovacích případů 6. Z konečného výsledku je patrné, že výsledný počet testovacích případů se snížil o 26.

Tab. č. 3.15 Výsledná pravdivostní tabulka

R (rozhodnutí)									
Správně					Chybně				
A	B	C	D	E	A	B	C	D	E
1	0	1	1	1	**0**	0	1	1	1
1	**1**	0	1	1	1	**0**	0	1	1
~~1~~	~~0~~	~~**1**~~	~~1~~	~~1~~	~~1~~	~~0~~	~~**0**~~	~~1~~	~~1~~
~~1~~	~~0~~	~~1~~	~~**1**~~	~~1~~	1	0	1	**0**	1
~~1~~	~~0~~	~~1~~	~~1~~	~~**1**~~	1	0	1	1	**0**

Konkrétní příklad – Zapnutí navigačního modulu



Obr.č. 3.6 Zapnutí navigačního modulu

$$On\ or\ (Radio\ and\ NAV) = Navigace\ zapnuta$$

Pokud bude klíček od automobilu v zapalování v pozici 2 (On) nebo alespoň v pozici 1 (Radio), bude aktivován navigační modul. Zadání příkladu je zobrazeno na obr.č. 3.6. V prvním kroku vytvoříme tabulku (tab. č. 3.16), do které nejprve zapíšeme rozhodovací diagonály. Ze zadání příkladu lze rozpoznat 3 podmínky (On, Radio, NAV) n = 3 a celkový počet testovacích případů je $2^3 = 8$.

Tab. č. 3.16 Pravdivostní tabulka

Navigace zapnuta					
Správně			Chybně		
On	Radio	NAV	On	Radio	NAV
1			**0**		
	1			**0**	
		1			**0**

Na základě rozhodovacích úrovní doplníme zbylé buňky tabulky (tab. č. 3.16) tak, aby odpovídala zadání příkladu. Zároveň musí být dodrženy všechny podmínky této metody. V případě záměny rozhodovací úrovně se musí změnit i konečný výsledek.

Tab. č. 3.17 Výsledná tabulka

Navigace zapnuta					
Správně			Chybně		
On	Radio	NAV	On	Radio	NAV
1	0	1	**0**	0	1
0	**1**	1	~~0~~	~~**0**~~	~~1~~
~~0~~	~~1~~	~~**1**~~	0	1	**0**

Z tab. č. 3.17 je patrné, že počet testovacích případů je 6 (> n + 1) a to neodpovídá výslednému efektu metody MCDC. Jsou-li některé řádky v tabulce stejné, můžeme je vyškrtnout a tím dosáhnout konečného optimálního výsledku – počet testovacích případů 4. Z konečného výsledku je patrné, že výsledný počet testovacích případů se snížil o 4.

3.2 Neformální techniky

V neformálních technikách je vysoký stupeň volnosti při vytváření testovacích případů. Odvození testovacích případů není nezávislé na testovacím inženýrovi. V těchto metodách je vyžadována jeho kreativita a odborná znalost. Požadavky na bázi podkladů pro testování jsou nižší než u formálních technik. Prohlášení o pokrytí testů založené na bázi podkladů pro testování mohou být přijata pouze podmínečně.

3.2.1 Klasifikační strom

Klasifikačních stromů je celá řada. Ukažme si jednoduchý příklad klasifikace. Představte si, že chcete vytvořit systém na třídění mincí a chcete je rozdělit do různých tříd (koruny, dvoukoruny, pětikoruny a desetikoruny). Mince se liší průměrem, na základě kterého bude navrhnut tento systém. Mince se pohybují po úzké dráze, ve které jsou vytvořeny otvory dle daných průměrů od nejmenší po největší. Jestliže mince spadne prvním otvorem, je klasifikována jako koruna, pokud ne pokračuje dále po této dráze dokud nespadne, a tak dále. Tomuto procesu se říká klasifikační strom. Rozhodovací proces, který jsme zde uvedli, poskytuje efektivní způsob třídění mincí. Obecně můžeme říci, že tento proces lze použít pro širokou škálu klasifikačních problémů. Klasifikační strom je dále popsán v [7].

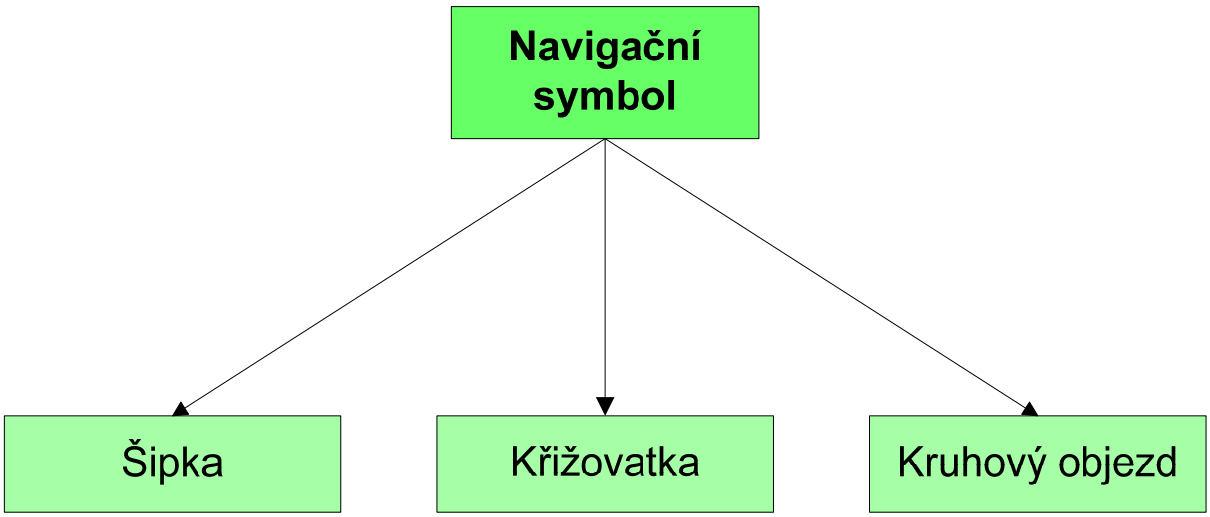
Klasifikační stromy se používají v širokém spektru činností jako například v diagnostice, počítačových vědách (datové struktury), botanice a psychologii (teorie rozhodování). Klasifikační stromy lze vyjádřit graficky, čímž je lze snadněji interpretovat, než když se jedná pouze o numerickou interpretaci.

Klasifikace touto metodou (Classification Tree Method, CTM) slouží k systematické analýze oblasti vstupních dat testovaného objektu. Vstupní oblast je klasifikována pod různými aspekty. Testovací případy jsou definovány na základě kombinace tříd. CTM je neformální Black-Box technika návrhu testovacích případů. Může být použita na různých úrovních testování softwaru. Výstupní dokumenty nevyžadují žádnou zvláštní formu.

Obecný postup:

1. Identifikovat parametry testovaného objektu.
2. Zařadit hodnoty parametrů do tříd ekvivalence.
3. Stanovit hodnoty pro každou kategorii.
4. Naplnit klasifikační strom kategoriemi a hodnotami.
5. Vytvořit logické testovací případy.
6. Definovat počáteční a ukončovací podmínky.
7. Vytvořit fyzikální testovací případy.

Základní myšlenkou metody klasifikačního stromu je nejprve rozdělit množství vstupů zkušebního objektu, které budou dále rozděleny různými způsoby, kde budou brány v úvahu všechny vhodné aspekty. Rozdělené vstupy budou následně kombinovány tak, aby vytvořily testovací případy, které pokryjí celou vstupní datovou doménu. V prvním kroku je nutné zavést aspekty, které jsou relevantní k testu. Každý aspekt by měl být úzce omezen, aby bylo jasné rozlišení mezi možnými vstupy pro zkušební objekt. V následujícím kroku je množství možných vstupů rozděleno podle jednotlivých aspektů. Toto rozdělení je nazýváno jako klasifikace. Ukázka možnosti zobrazení klasifikačního stromu je znázorněna na obr.č. 3.7.



Obr.č. 3.7 Ukázka klasifikačního stromu

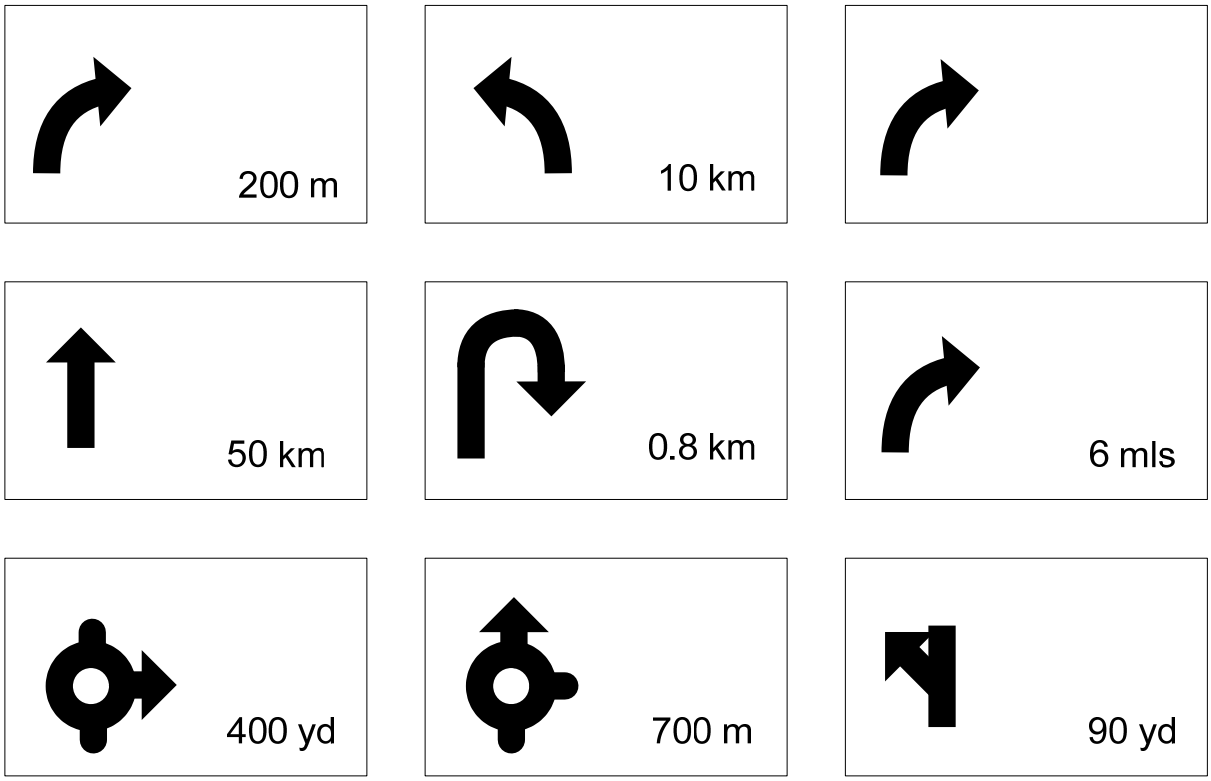
3.2.2 Analýza hraničních hodnot

Jedná se o zvláštní případ metody Třídy ekvivalence. Tato metoda vznikla na základě dlouhodobého pozorování výskytu chyb. Nejčastěji se chyby objevují na hranici tříd ekvivalence. Jedná se o testování tzv. okrajových podmínek. Pro analýzu hraničních hodnot není žádný nástroj.

Symboly navigace

Na následujícím obrázku (obr.č. 3.8) jsou uvedeny symboly navigace. Hodnoty, které jsou zaznamenány navigací, jsou rozděleny do následujících tříd ekvivalence:

- Ukazatel směru: šipka, křižovatka, kruhový objezd
- Úhel symbolu: 0°, 30°, 45°, 90°, 135°, 180°, 225°, 270°, 315°, 330°
- Grafický ukazatel vzdálenosti: 0% - 100% (krok 25%)
- Numerický ukazatel vzdálenosti: zobrazený, skrytý
- Hodnoty numerického ukazatele: 1 – 9
- Rozlišení numerického ukazatele: 1, 10, 100, 0.1
- Jednotka numerického ukazatele: km, m, mls, yd



Obr.č. 3.8 Symboly navigace

Hraniční hodnoty

- Třída: Úhel symbolu jízdního pokynu
    - Binární oblast hodnot: 10 bit (0-1023)
    - Fyzikální oblast hodnot: 0-359,5°
    - Rozlišení: 0,5°
- Hodnota pro dobrý případ (1 test na požadavek): 90° →1 hodnota
- Všechny směry šipek (Klasifikace): 0°, 30°, 45°, 90° →10 hodnot
- Analýza hraničních hodnot: 0°, 14°, 14.5°, 15°, 15.5°, 30°, 37°,… →14-50 hodnot
- Dovolené hodnoty: 0°, 0.5°, 1°, 1.5°, 2°, 2.5°, … →720 hodnot
- Celková oblast hodnot: 0°, 0.5°, …511.5° →1024 hodnot
- Hraniční oblast hodnot: 0°, 0.5°, 359°, 359.5°, →4 hodnoty
- Chybné případy: 360°, 511.5°, 512°, 800° →4 hodnoty

3.2.3 Dobrý případ (něm. Gutfall)

Metoda Gutfall se zabývá testováním všech dobrých případů. Tato metoda patří mezi neformální techniky specifikace testů. Jedná se o základní testování funkčnosti produktu při standardních podmínkách. Pro metodu Gutfall neexistuje žádný nástroj pro vytvoření specifikace testu.

3.2.4 Odhadování chyby (angl. Error Guessing)

Metoda „Odhadování chyby“ není sama o sobě testovací technika, ale spíše dovednost, která může být aplikována na všechny ostatní testovací techniky ke stanovení účinnějších testů, tj. testů, které najdou co nejvíce chyb. Tato metoda umožňuje najít chyby nebo nedostatky v návrhu testů. Ve skutečnosti je možné využívat řadu technik včetně:

- znalostí o návrhu testů, např. metody návrhu nebo implementace technologie,
- znalost výsledků dřívějších testovacích fází,
- zkušenosti z testování podobných nebo souvisejících systémů (např. vědět, kde se závady objevily již v dříve testovaných systémech),
- znalost typických implementačních chyb (např. dělení nulou),
- obecná pravidla testování – heuristika (přibližné řešení).

Již ze samotného názvu je zřejmé, že základem této metody je odhad, kde mohou být skryté chyby. Neexistují žádné specifické nástroje a techniky. Metoda „Odhadování chyby“ nemá přímo stanovená pravidla pro testování. Testovací případy mohou být navrženy v závislosti na situaci buď z funkčních dokumentů, nebo pokud jsou zjištěny neočekávané / nedoložené chyby při samotném testování.

Odhadování chyby je spíše dovednost, kterou stojí za to pěstovat, protože může dělat testování mnohem efektivnější a účinnější. Tato dovednost odhadování chyb přichází se zkušenostmi z předchozích projektů a testování.

Někteří lidé mají přirozený talent pro programové testování, ačkoli často nepoužívají žádné formální techniky, ale používají intuitivně metodu „Odhadování chyby“ (ang. Error Guessing) [8].

Příklad „Stop – Start“ (např. při zastavení na semaforu)

Při rychlosti pod 3 km/h se motor automaticky zastaví, když je pedál spojky uvolněn. Předpokladem pro to je volnoběh (řadicí páka – pozice N) a stisknutý brzdový pedál.

Logika (bez časových záznamů):

$$\left(v < 3\frac{km}{h}\right) and (stupeň = N) and (brzda = stisknuta) and (spojka = nestisknuta)$$
$$\rightarrow motor\ vypnut$$

Testovací případy:

1. (v<3 km/h) and (N) and (brzda) and (bez spojky) → Motor neběží,
2. (v>3 km/h) and (N) and (brzda) and (bez spojky) → Motor běží,
3. (v<3 km/h) and (1) and (brzda) and (bez spojky) → Motor běží,
4. (v<3 km/h) and (N) and (bez brzdy) and (bez spojky) → Motor běží,
5. (v<3 km/h) and (N) and (brzda) and (spojka) → Motor běží,
6. Test hraniční hodnoty v=3 km/h, (dále jako 1. a 2.),
7. Přezkoušení časové souvislosti (jiná pořadí),
8. Dodržení dalších počátečních podmínek (např. funkce deaktivovat manuálně).

Použití metod:

- Pro 1. - 5. bod bylo možné použít určení tříd ekvivalence nebo MCDC.
- Pro bod číslo 6 byla aplikována Analýza hraničních hodnot.
- Pro 7. a 8. bod byla použita metoda Odhad chyby.

4 Nástroje pro specifikaci testů

Pro většinu uvedených metod nejsou dostupné žádné nástroje pro specifikaci testů. Testovací případy se vytvářejí ručně a to buď formální technikou nebo neformální technikou. Pro vytvoření testovacích případů pomocí metody klasifikačního stromu lze použít bezplatně dostupnou základní verzi programu CTE XL od společnosti Berner & Mattner.

4.1 CTE XL

V 90. letech byla nejdříve vyvinuta metoda klasifikační strom a následně byl vytvořen editor pro tvorbu klasifikačních stromů. Kontextový editor je výkonný nástroj pro systematickou specifikaci a implementaci testovacích případů. CTE je oborově nezávislý, je vhodný při vývoji systému, ale taktéž při vývoji softwaru. Spektrum použití sahá od požadavků na systémovou analýzu, až po implementaci testovacích případů. Specifikace testu jsou vyvinuty ve vyšší kvalitě a za kratší dobu [9].

CTE (Classification Tree Editor) neboli v českém znění grafický editor pro tvorbu klasifikačních stromů byl vytvořen společností Berner & Mattner. Jedná se o základní verzi s omezenými funkcemi, která je zdarma ke stažení na stránkách této společnosti [9]. Cena plné verze programu CTE XL Professional se pohybuje okolo 30 000 Kč, ve které je již zahrnuta i podpora tohoto programu. Kontextový editor je založen na osvědčeném způsobu klasifikace stromů. Tento nástroj je i přes jeho jednoduchou použitelnost výjimečně užitečný při analýze systémového prostředí a stanovení zkušebních oblastí. CTE je běžně používán ve vývojovém a testovacím prostředí a dále se velmi úspěšně používá v různých dalších oblastech [9].

Vytvoření testovacího případu je testovací činnost, která je rozhodující pro kvalitu testu, protože výběr testovacích případů stanovuje povahu a rozsah zkoušky. Pokud jsou testovací případy stanoveny na základě specifikace, nazývají se jako funkční testy. Ačkoli mají funkční testy velký význam pro ověření systémů a běžně se používají v průmyslu, jen pár metod a nástrojů je k dispozici pro systematické vytváření odpovídajících testovacích případů. Metoda klasifikační strom umožňuje rozdělení celé vstupní oblasti zkušebního objektu do samostatných tříd prostřednictvím klasifikací. Nástroj CTE byl vytvořen pro usnadnění tvorby klasifikačního stromu, a tak aby aplikace této metody byla co nejefektivnější [9].

4.1.1 Základy programu CTE XL

Pro vytvoření klasifikačního stromu je nutné znát základní popis prvků, které budou použity, a které prvky mohou být propojeny.

Popis prvků

- **Kořenový element** – Tento prvek je kořenem stromu a nelze ho odstranit. Představuje objekt, který má být testován. Klasifikace a kompozice mohou být vytvořeny v rámci kořenového elementu.
- **Klasifikace** – Klasifikace jsou aspekty, podle kterých je objekt klasifikován, např. barva, tvar atd. Na základě klasifikace mohou být vytvořeny třídy.
- **Třídy** – Třídy jsou charakteristiky testovacích aspektů. Třídy mohou být propojeny do testovacích případů v kombinační tabulce. Klasifikace a kompozice mohou být vytvořeny v rámci třídy.
- **Kompozice** – Kompozice může shrnout několik klasifikací. Třídy několika klasifikací, které jsou umístěny v kompozici, se vzájemně nevylučují. Klasifikace a další kompozice mohou být vytvořeny pod kompozicí.

Propojení prvků

V tab. č. 4.1 lze zjistit, které prvky mohou být propojeny a vytvořeny jako podřízený element daného typu prvku.

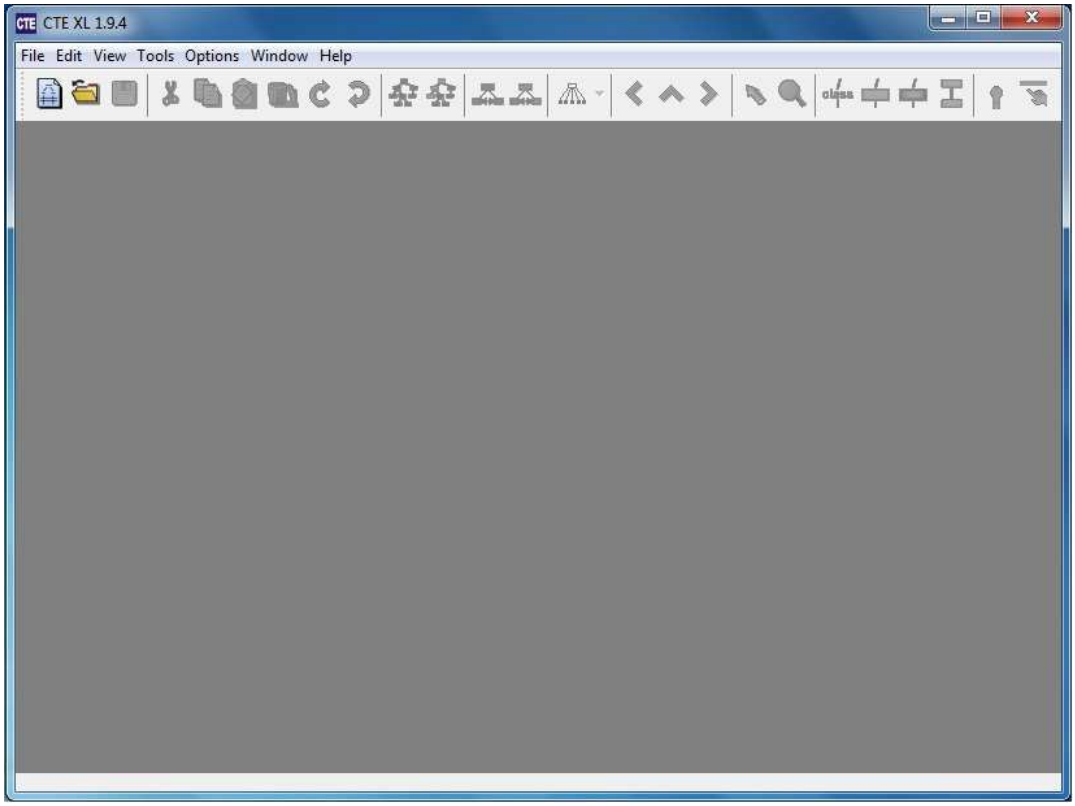
Tab. č. 4.1 Propojení prvků

Prvky / Prvky	Kořenový element	Klasifikace	Kompozice	Třída
Klasifikace	ANO	NE	ANO	ANO
Kompozice	ANO	NE	ANO	ANO
Třída	NE	ANO	NE	ANO

Existují dva způsoby vytváření klasifikačních stromů. Můžeme buď stavět od kořene směrem nahoru (top-down design) a nebo vytvoříme všechny prvky nezávisle a potom je připojíme (bottom-up design).

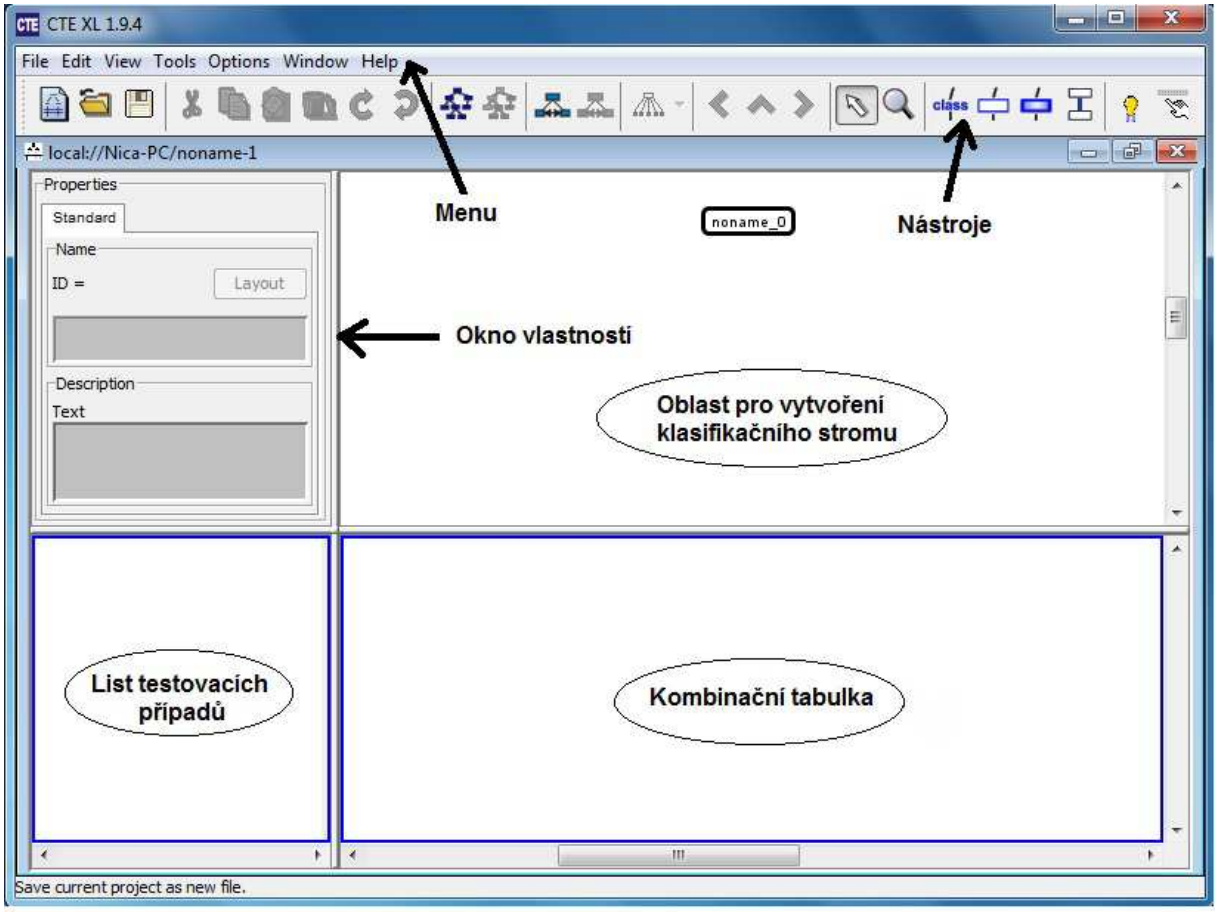
4.1.2 Vytvoření klasifikačního stromu

Nejprve si nainstalujeme program CTE XL. Při prvním spuštění je nutná registrace jak pro spuštění bezplatné verze tak i pro placenou verzi. V nabídce Options lze zvolit jazyk – německý nebo anglický. Hlavní okno CTE XL lze vidět na obr.č. 4.1.



Obr.č. 4.1 Hlavní okno programu CTE XL

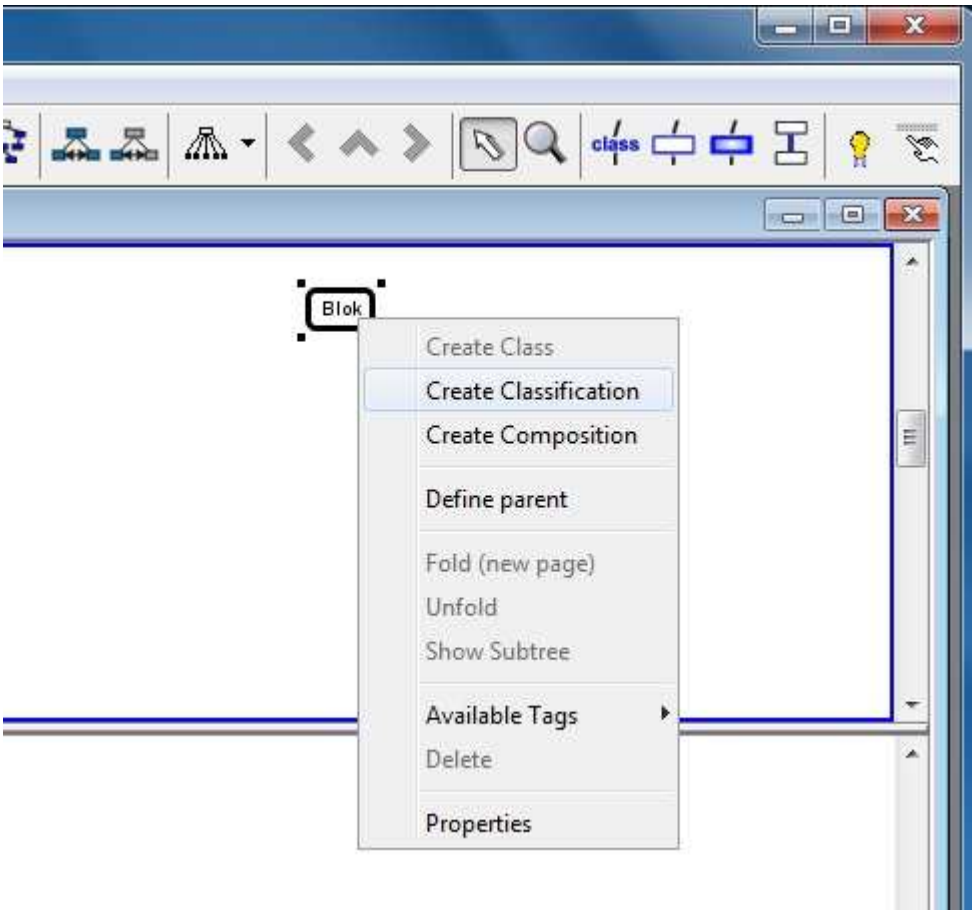
Nový dokument vytvoříme kliknutím na tlačítko File (Soubor) a zde vybereme New (Nový) nebo můžeme použít klávesovou zkratku Ctrl + N. Objeví se nové podokno, které obsahuje prázdný dokument. Toto podokno, které je znázorněno na obr.č. 4.2, je rozděleno do čtyř oblastí. V pravém horním rohu se nachází oblast pro vytvoření klasifikačního stromu, která již obsahuje kořenový element s názvem noname_0, což je výchozí bod pro klasifikační strom. Pod touto oblastí je umístěna kombinační tabulka, ve které lze propojit testovací případy s prvky stromu. Na levé straně bude zobrazen list testovacích případů. Plocha pro vlastnosti aktuálně vybraného prvku lze upravovat v oblasti, která je v levém horním rohu.



Obr.č. 4.2 Prázdný dokument

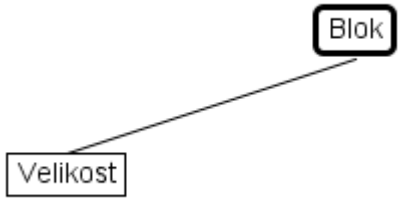
Kořenový element můžeme přejmenovat dvojitým kliknutím levým tlačítkem myši v okně klasifikace stromu. Nebo levým tlačítkem myši klikneme na kořenový element a přepneme se do okna vlastností, kde přepíšeme nový název (např. blok) do horního textového pole.

A nyní můžeme rozvíjet klasifikační strom. Pro znázornění vytvoříme 3 klasifikace (velikost, barva, tvar). Klikneme pravým tlačítkem myši na kořenový prvek a z kontextového menu zvolíme možnost Create Classification (vytvořit klasifikaci), tak jak je znázorněno na obr.č. 4.3.

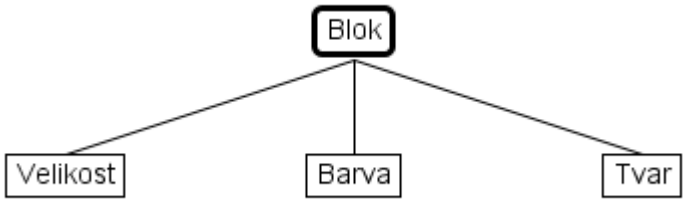


Obr.č. 4.3 Kořenový element a klasifikace

Nový prvek (noname_1) se zobrazí v okně pro tvorbu klasifikačního stromu. Kliknutím levým tlačítkem myši označíme prvek. Vlastnosti tohoto pole se objeví v levém horním bloku a zde přepíšeme název na *Velikost* (obr.č. 4.4).



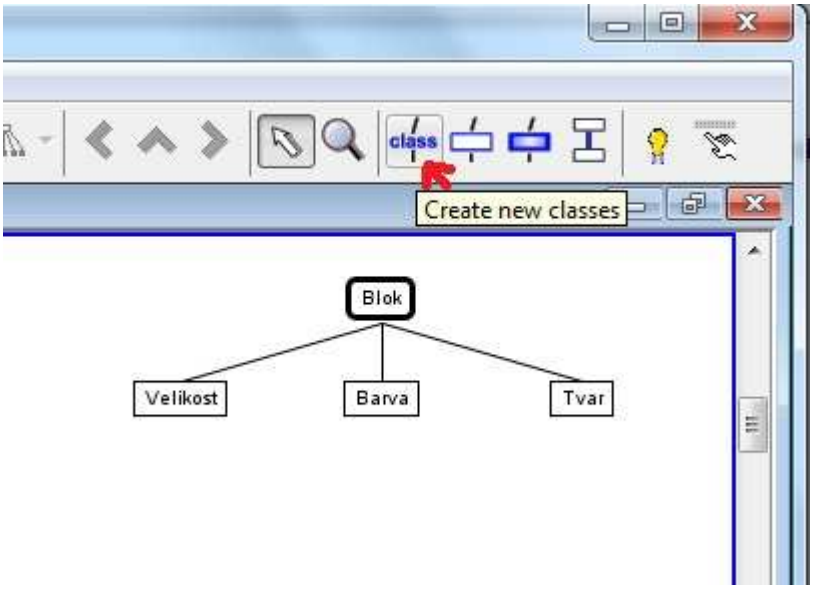
Obr.č. 4.4 Vytvoření klasifikace



Obr.č. 4.5 Klasifikační strom - ukázka

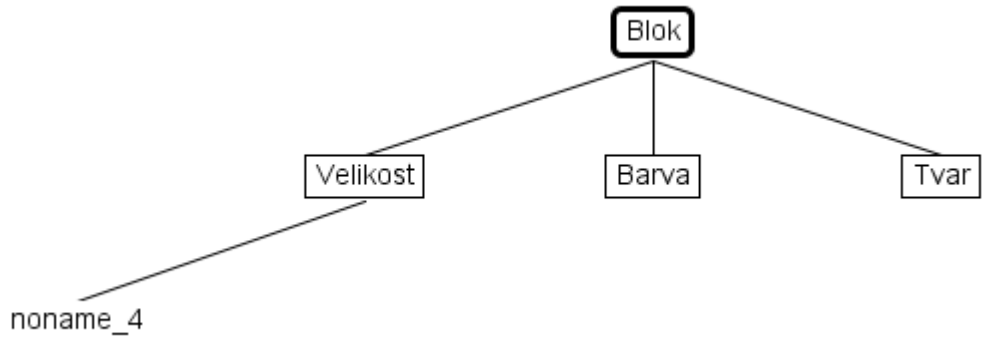
Další klasifikaci barvu a tvar vytvoříme stejným způsobem a výsledek je znázorněn na obr.č. 4.5.

Třídy mohou být vytvořeny stejným způsobem jako klasifikace, ale ukážeme si další možný způsob generování prvků. Klikneme levým tlačítkem myši na symbol Create new classes (Vytvoření nové třídy) v panelu nástrojů (obr.č. 4.6).



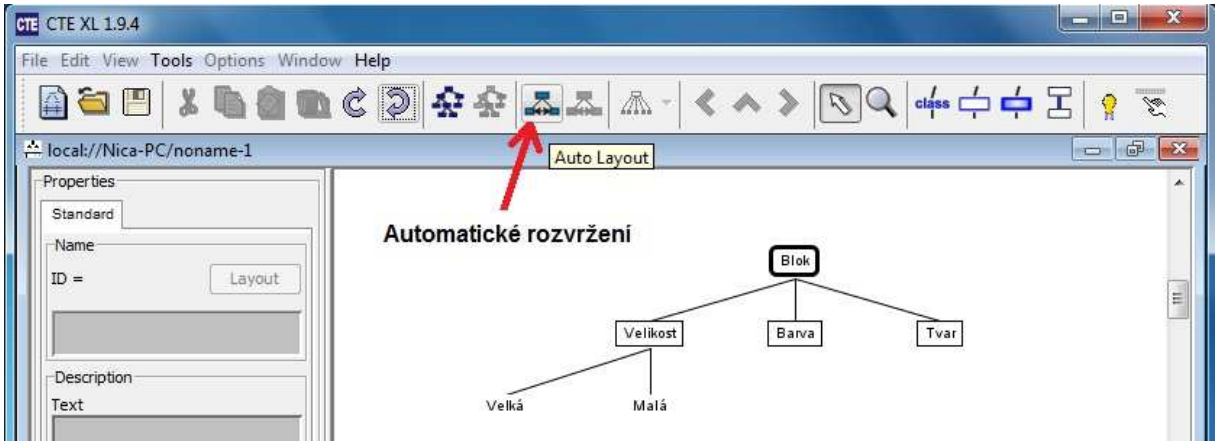
Obr.č. 4.6 Vytvoření třídy

Kurzor myši se změní a pokud klikneme na prvek klasifikace – Velikost, bude vytvořena nová třída na základě této klasifikace (obr.č. 4.7).



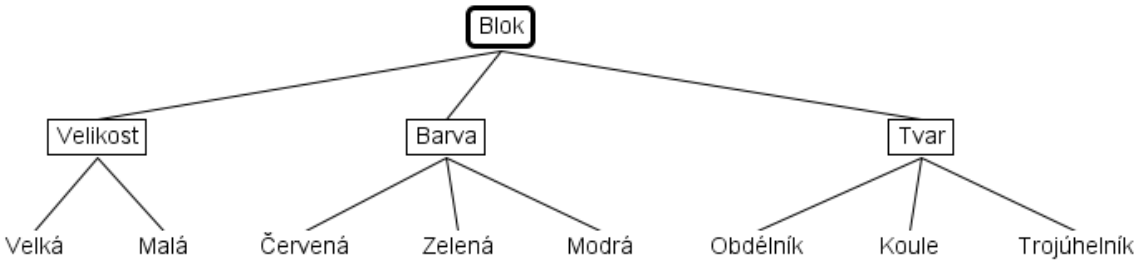
Obr.č. 4.7 Vytvoření třídy – ke klasifikaci Velikost

V rámci klasifikace *Velikost* vytvoříme stejným způsobem další třídu. Nyní se vrátíme zpět do normálního režimu zobrazení a to provedeme buď kliknutím pravého tlačítka myši do kreslicí plochy a nebo kliknutím na symbol ukazatele (šipky) v panelu nástrojů. Přejmenujeme nově vytvořené třídy dvojitým kliknutím na příslušnou třídu a zadáme nový název (Velká, Malá). Nyní přejdeme zpět do režimu pro tvoření tříd a vytvoříme další třídy u klasifikace Barev (Červená, Modrá a Zelená) stejným způsobem. U poslední klasifikace *Tvar* vytvoříme také 3 třídy (Obdélník, Koule a Trojúhelník). Pro seskupení prvků v klasifikačním stromu, lze spustit automatické rozvržení tlačítkem, které je zobrazeno na obr.č. 4.8.



Obr.č. 4.8 Automatické rozvržení

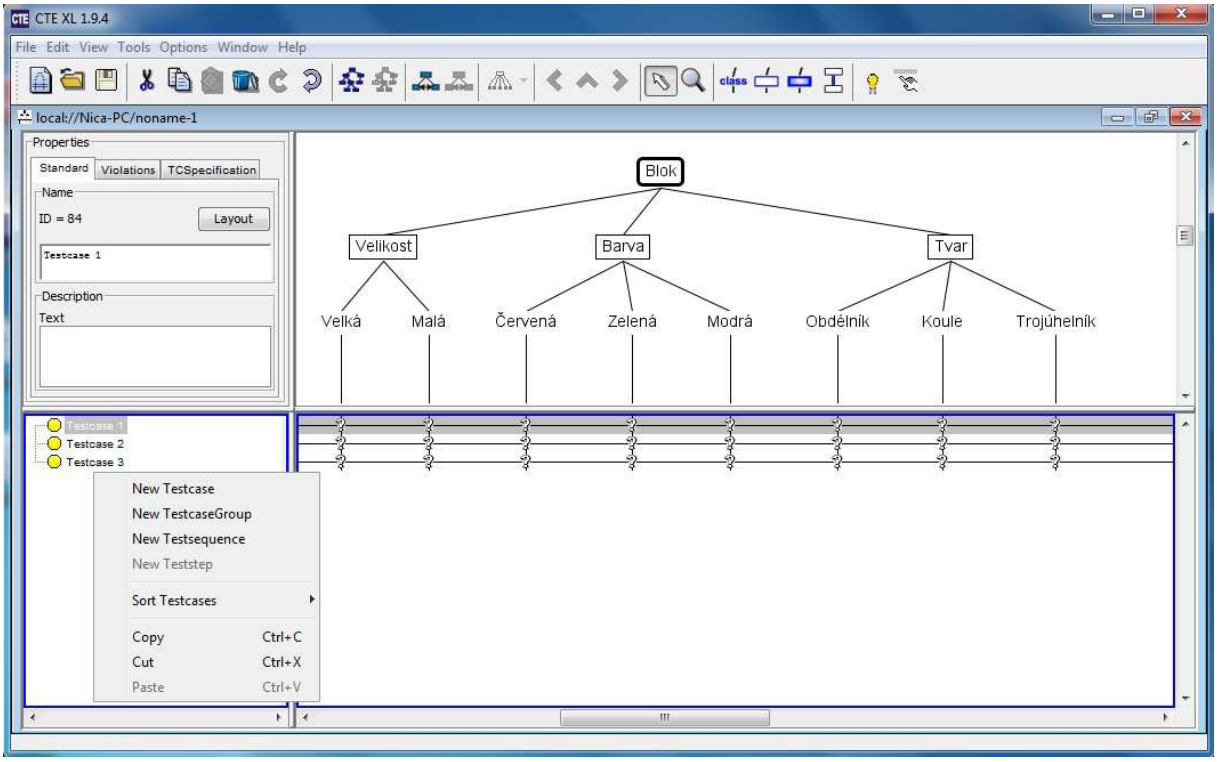
Po automatickém rozvržení nám vznikne výsledný klasifikační strom, který je znázorněn na obr.č. 4.9.



Obr.č. 4.9 Výsledný klasifikační strom

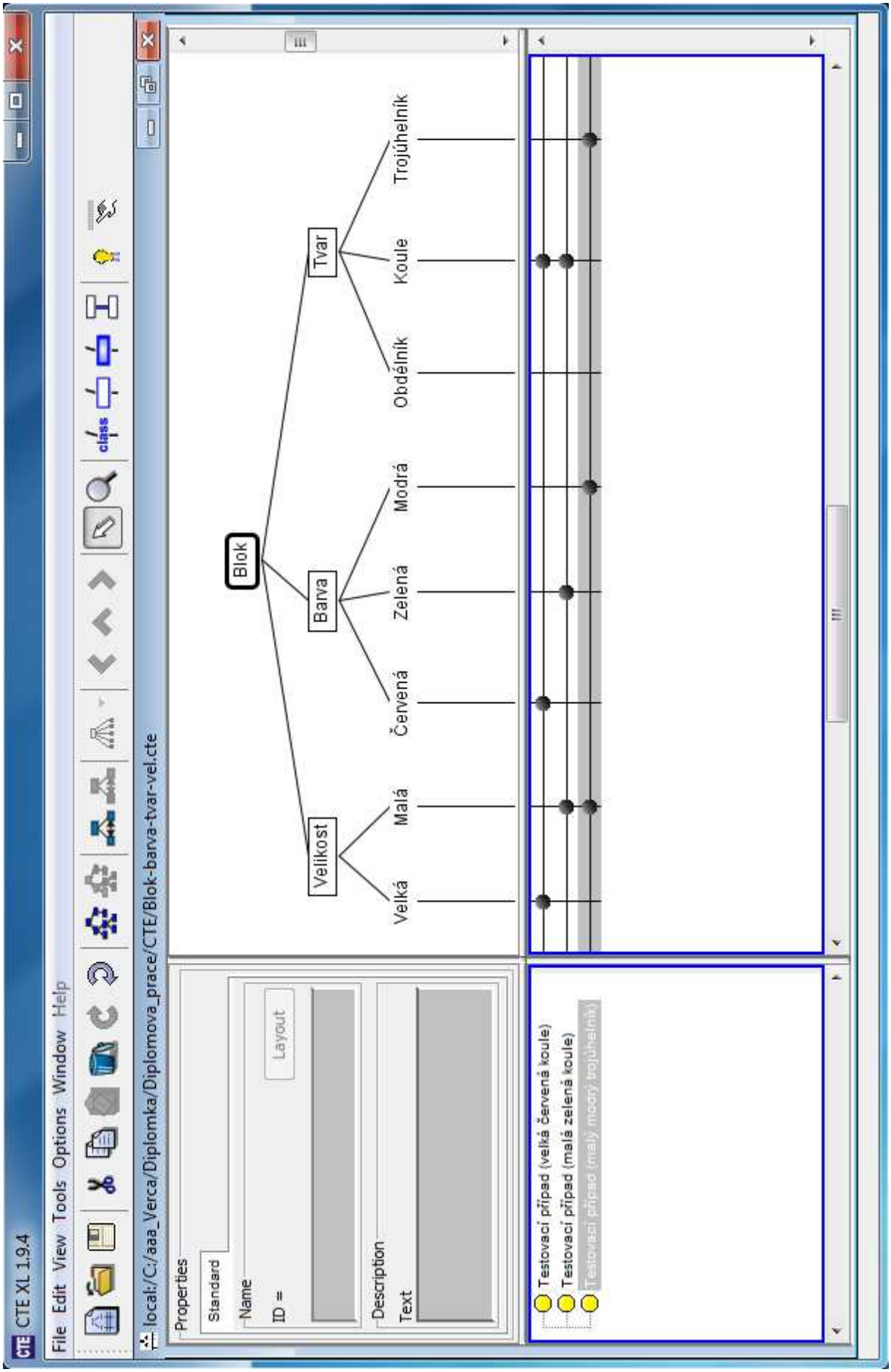
4.1.3 Vytvoření testovacích případů

V případě, že již máme vytvořen kompletní klasifikační strom, můžeme vytvářet jednotlivé testovací případy znázorněné na obr.č. 4.10. Klikneme pravým tlačítkem myši v seznamu testovacích případů (levá dolní oblast hlavního okna) a z kontextového menu vybereme New Testcase (Nový testovací případ). Tímto způsobem můžeme vytvořit další testovací případy.



Obr.č. 4.10 Generování testovacích případů

Po vygenerování testovacích případů se nám objevily nové řádky a sloupce v kombinační tabulce. Pomocí levého tlačítka myši můžeme provést přiřazení testovacích případů označením příslušného otazníku. Při výběru se ukazatel myši změní na velký černý bod a otazníky v dané klasifikaci zmizí, protože jeden prvek je již vybrán. V našem příkladu bude prvním testovacím případem výběr – malá červená koule. Stejným způsobem dokončíme kombinaci prvků zbylých dvou testovacích případů. Druhý testovacím případ bude – malá zelená koule a třetím případem bude – malý modrý trojúhelník. Konečný výsledek je zobrazen na obr.č. 4.11.



Obr.č. 4.11 Konečný výsledek – vygenerované testovací případy

5 Vygenerovaný test – ukázka

Pro znázornění, jak vytvořit testovací případy pomocí metody klasifikačního stromu, si předvedeme na příkladu, který byl zpracován metodou třídy ekvivalence – potkávací světlo (v kapitole č. 3.1.1).

Nejprve vytvoříme klasifikační strom pomocí programu CTE XL (Editor pro tvorbu klasifikačních stromů). Ten nakreslíme dle popisu uvedeného v kap. 4.1.2. Dále na základě klasifikačním stromu budou vytvořeny testovací případy, kde bude vidět i očekávaný výsledek (Výsledek – ON nebo OFF). Testovací případy:

- ***Testovací případ 1***

V prvním testovacím případu označíme Vypínač světel – Zapnuto, Světelný senzor – Světlo, Dešťový senzor – Sucho a Zapalování - Zapnuto. Očekávaný výsledek bude On (Potkávací světla budou zapnuta).

- ***Testovací případ 2***

Druhý případ zahrnuje: Vypínač světel – Vypnuto, Světelný senzor – Tma, Dešťový senzor – Mokro a Zapalování - Start. Očekávaný výsledek bude Off (Potkávací světla budou vypnuta).

- ***Testovací případ 3***

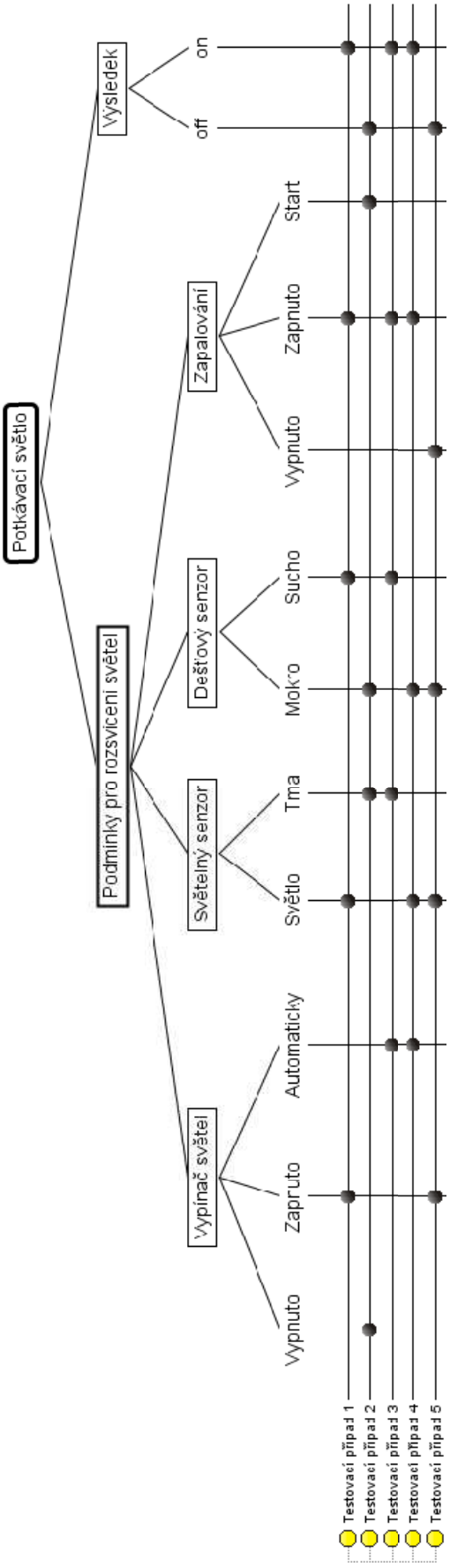
V třetím testovacím případu vybereme: Vypínač světel – Automaticky, Světelný senzor – Tma, Dešťový senzor – Sucho a Zapalování - Zapnuto. V tomto případu bude očekávaný výsledek On (Potkávací světla budou zapnuta).

- ***Testovací případ 4***

Ve čtvrtém testovacím případu označíme: Vypínač světel – Automaticky, Světelný senzor – Světlo, Dešťový senzor – Mokro a Zapalování - Zapnuto. V tomto případu bude očekávaný výsledek On (Potkávací světla budou zapnuta).

- ***Testovací případ 5***

Pátý testovací případ bude obsahovat: Vypínač světel – Zapnuto, Světelný senzor – Světlo, Dešťový senzor – Mokro a Zapalování - Vypnuto. Očekávaný výsledek bude Off (Potkávací světla budou vypnuta).



Obr.č. 5.1 Klasifikační strom a testovací případy – Potkávací světlo

6 Závěr

Testování je dnes nedílnou součástí vývojového procesu a to již od samého začátku. V případě vývojových procesů, které ke své práci používají pokročilý model – např. V-model, je kladen požadavek na opakovatelnost prováděných testů. K vytvoření testovacích případů jsou je zpravidla nutné použít metody pro specifikaci testů a samozřejmě i určité programové vybavení.

Úvodní část byla věnována obecnému pojmu testování. V kapitole 2.4 je důkladně popsán testovací proces, který se skládá z pěti částí – strategie testovací kampaně, plánování testovací kampaně, specifikace testů, realizace testů a vyhodnocení testů. V této části je také zmíněn velice často používaný V-model vývojového procesu. Dalším bodem byl popsán princip a implementace HiL testování.

Předmětem této práce bylo sepsání nejčastěji používaných metod pro specifikaci testů. Techniky specifikace mohou být formální nebo neformální. Mezi formální techniky patří metoda Tříd ekvivalence a MCDC. Mezi neformální techniky přísluší metody: Klasifikační strom, Analýza hraničních hodnot, Gutfall a Error Guessing.

Výsledkem této práce bylo nalezení vhodného nástroje, pomocí kterého by bylo možné vytvořit testovací případy. Pro vytvoření testovacích případů byl nalezen nástroj pro tvorbu klasifikačních stromů. Jedná se o volně dostupnou základní verzi programu CTE XL od společnosti Berner & Mattner, která má omezené funkce. V poslední části je vygenerována ukázka testovacích případů pomocí tohoto nástroje.

7 Zdroje a literatura

[1] SAX (HRSG.), Eric. *Automatisiertes Testen Eingebetteter Systeme in der Automobilindustrie,*. München: Hanser Fachbuchverlag, 2008. ISBN 978-3-446-41635-2.

[2] PDQM. *PDQM - Less Management Work* [online]. 2012 [cit. 2012-03-20]. Dostupné z: http://www.pdqm.cz/Standards/SPICE.html

[3] International Organization for Standardization. *International Organization for Standardization* [online]. 2011 [cit. 2012-03-25]. Dostupné z: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51356

[4] Kubík, Michal. *Testování elektronických systémů automobilu*. Plzeň, 2011. Disertační práce. Západočeská univerzita v Plzni. Fakulta elektrotechnická. Školitel prof. Ing. Jiří Pinker, CSc.

[5] Testování softwaru. *Testování softwaru* [online]. 2010 [cit. 2012-04-08]. Dostupné z: http://testovanisoftwaru.cz/dokumentace-v-testovani/test-case

[6] Hayhurst, Kelly – Veerhusen, Dan – Chilenski, John – Rierson, Leanna. *A Practical Tutorial on Modified Condition/ Decision Coverage*. [online] Hampton (Virginia, USA): NASA, 2001. NASA/TM-2001-210876. [cit.: 2012-04-14]. Dostupné z: <http://shemesh.larc.nasa.gov/fm/papers/Hayhurst-2001-tm210876-MCDC.pdf>

[7] StatSoft. *StatStoft Electronic Statistics Textbook* [online]. 2011 [cit. 2012-04-16]. Dostupné z: http://www.statsoft.com/textbook/classification-trees/

[8] Software Testing. *Software Testing* [online]. 2007 [cit. 2012-04-18]. Dostupné z: http://amit-badola.blogspot.com/2007/10/error-guessing.html

[9] Berner & Mattner. *Berner & Mattner* [online]. 2011 [cit. 2012-04-21]. Dostupné z: http://www.berner-mattner.com

Dále byly jako zdroje informací použity interní materiály společnosti MBtech Bohemia a příručka k programu CTE XL.