VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

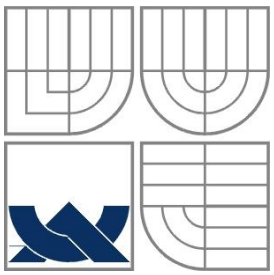
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

PROBLEMATIKA TESTOVÁNÍ A VERIFIKACE SOFTWARU PRO LETECKOU TECHNIKU

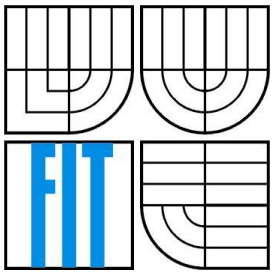
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Lukáš Mačišák
AUTHOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

PROBLEMATIKA TESTOVÁNÍ A VERIFIKACE SOFTWARU PRO LETECKOU TECHNIKU

METHODS OF SOFTWARE TESTING AND VERIFICATION FOR AIRBORNE SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Lukáš Mačišák
AUTHOR

VEDOUCÍ PRÁCE Doc. Ing., Dipl.-Ing. Martin Drahanský, Ph.D.
SUPERVISOR

BRNO 2011

Abstrakt

Tato diplomová práce popisuje problematiku certifikace a vývoje softwaru pro leteckou techniku se zaměřením na oblast testování a verifikace v rámci životního cyklu projektu. Práce obsahuje navrhnutý vlastní postup verifikace softwaru ve shodě s požadavky normy RTCA/DO-178B pro konkrétní aplikaci. Součásti je i ukázková realizace takových částí navrhnutého postupu, které se týkají testování softwaru. V závěru jsou popsané možnosti uplatnění navrženého postupu a zhodnocení jeho výsledků.

Abstract

This Master‘s Thesis describes methods of software certification and development of airborne systems, focusing on software testing and verification during project’s life cycle. Thesis includes also designed software verification plan for concrete application according to RTCA/DO-178B. Another part of thesis illustrates the exemplary realization of tests according to designed verification plan. At the close we describe the options of applying the designed verification plan and evaluation of its results.

Klíčová slova

Certifikace softwaru, verifikace softwaru, kritický software, letecký software, avionické systémy, testovací metody a postupy, integrační testování, modulární testování, systémové testování, Cantata++, DO-178B.

Keywords

Software certification, software verification, airborne software, avionics systems, methods of testing, integration tests, modular tests, system tests, Cantata++, DO-178B.

Citace

Mačišák Lukáš: Problematika testování a verifikace softwaru pro leteckou techniku, semestrální projekt, Brno, FIT VUT v Brně, 2011

Problematika testování a verifikace softwaru pro leteckou techniku

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením doc. Martina Drahanského.
Další informace mi poskytl Ing. David Svačina.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Lukáš Mačišák
23.5.2011

Poděkování

Chcel by som poďakovať vedúcemu práce doc. Martinovi Drahanskému a konzultantovi Ing. Davidovi Svačinovi za ich užitočné rady a pripomienky. Chcem sa poďakovať firme UNIS a.s. za poskytnutie hardwaru, softwaru a dokumentov potrebných na tvorbu tejto práce.

Použité skratky

SW – Software

HW – Hardware

ZRM – Zariadenie pre riadenie chodu prúdového motoru

TPZRM – Testovací prípravok pre ZRM

PACS – Plán aspektov certifikácie SW

PZK – Plán zabezpečenia kvality

PKM – Plán konfiguračného manažmentu

PVS – Plán vývoja SW

VPS – Verifikačný plán SW

ŠPP – Štandard pre požiadavky

ŠPN – Štandard pre návrh

ŠPK – Štandard pre kódovanie

VÚP – Vysokoúrovňové požiadavky na SW

NÚP – Nízkoúrovňové požiadavky na SW

TS – Technická špecifikácia

Obsah

1 Úvod

Táto diplomová práca sa zaoberá problematikou testovania a verifikácie softwaru pre leteckú techniku. Diplomová práca vznikla v spolupráci s firmou UNIS a.s. Tému diplomovej práce som si vybral na základe osobného záujmu o vývoj leteckého softwaru. V rámci pracovnej náplne sa podieľam sa na vývoji a certifikácii takýchto aplikácii. Keďže z vlastných skúsenosti viem, že táto problematika je veľmi rozsiahla a náročná na štúdium, výstupom práce by mala byť názorná ukážka priebehu verifikácie softwaru pre leteckú techniku. Druhá, tretia a štvrtá kapitola vychádzajú z poznatkov nadobudnutých pri riešení semestrálneho projektu.

Druhá kapitola sa zaoberá všeobecným pohľadom na životne kritický software a životne kritický systém, taktiež sa venuje porovnaniu rôznych štandardov pre certifikáciu systému v rôznych odvetviach priemyslu.

Úvod tretej kapitoly je venovaný certifikačným organizáciám. Hlavnú časť tejto kapitoly tvorí popis požiadaviek certifikácie a vývoja softwaru podľa normy DO-178B. V tomto oddiele sú popísané systémové aspekty, ktoré je nutné pochopiť pre následné pochopenie životného cyklu softwaru. Ďalej je tu popísaný životný cyklus softwaru so zameraním na procesy zaoberajúce sa testovaním a verifikáciou.

Štvrtá kapitola sa venuje vypracovaniu postupu verifikácie softwaru určeného pre konkrétny systém. Obsahuje podrobne rozpísané jednotlivé body verifikačného plánu, a to organizácia ľudí, pravidlá potrebnej nezávislosti pri verifikačných procesoch, metódy preskúmania, analytické a skúšobné metódy, verifikačné aktivity a overovacie prostredie.

Piata kapitola obsahuje popis použitia vytvoreného postupu verifikácie v praxi. Súčasťou tejto kapitoly je implementácia a výsledky systémového, modulárneho a integračného testovania a výsledky statickej analýzy. Záver tejto kapitoly tvorí celkové zhodnotenie navrhnutého postupu verifikácie a jeho praktickej realizácie.

Posledná, šiesta kapitola obsahuje záver, ktorý rozoberá dosiahnuté výsledky, prínos diplomovej práce a splnenie zadania.

2 Životne kritický software

Pre väčšinu firiem, vstupom do 21. storočia, sa SW stal dôležitou súčasťou ich každodenného pracovného prostredia, podnikania a produktov. Avšak, ak tento SW zlyhá, podniku vznikajú straty či už cez stratenú produktivitu pracovníkov, ušlý zisk, stratu potenciálu predaja, stratu zákazníkov, stratu alebo poškodenie dát, stratu na nákladoch potrebných k zálohe systému, k jeho obnoveniu, k rekonštrukcii jeho dát a pod. Napriek tomu, že sú firmy enormne závislé na SW, majú malý záujem o hodnotenie jeho kvality, rozhodujú sa na základe:

- Dobrej povesti predávajúceho.
- Marketingovej stratégie predávajúceho.
- Neoficiálnych odporúčaní od zamestnancov (kolegov).
- Publikovaných recenzií.

Podniky sa zriedkavo zaoberajú kvalitou SW, väčšinou ani nemôžu, a to z dôvodu nedostatku času a aj zdrojov na jeho testovanie. Z týchto dôvodov vzniká potreba „tretej strany“ (spoločnosť, organizácia), ktorá vytvorí nezávislé hodnotenia kvality.

Certifikácia SW môže byť účinným nástrojom pre spotrebiteľa, ktorým vie zistiť spoľahlivosť SW v rôznych konfiguráciách a prostrediach. Certifikát pre podnik bude taktiež slúžiť ako silný marketingový nástroj na odlíšenie sa od konkurencie. Certifikácia je nutnou súčasťou väčšiny životne kritických systémov. Tieto a ďalšie informácie sú uvedené v [5].

Táto práca sa zaoberá práve životne kritickým SW.

2.1 Životne kritický systém

Životne kritický systém alebo bezpečnostne kritický systém je systém, ktorého zlyhanie alebo porucha môže spôsobiť:

- Smrť alebo vážne poranenia ľudí.
- Stratu alebo závažné poškodenie drahého zariadenia.
- Vážne poškodenie životného prostredia.

Riziká tohto druhu sú zvyčajne zvládnuté pomocou metód a nástrojov bezpečnostného inžinierstva. Životne kritický systém by mal byť navrhnutý tak, aby spôsobil stratu menšiu než jeden ľudský život za miliardu prevádzkových hodín [7]. Typické návrhové metódy zahrňujú PRA (probabilistic risk assessment – pravdepodobnostný odhad rizík) a kombinujú FMEA (A failure modes and effects analysis – módy zlyhania a analýza ich účinkov) s FTA (fault tree analysis – analýzy pomocou stromu zlyhaní – testujú sa rôzne kombinácie zlyhaní). Tieto a ďalšie informácie je možné nájsť aj v [7].

Každý takýto systém musí spĺňať základné požiadavky, a to:

- Zaistenie bezpečnosti.
- Zabezpečenie robustnosi a spoľahlivosti SW.

- Ochrana ľudí musí byť uprednostňovaná pred ochranou hmotných vecí.
- Pri návrhu systému je potrebné brať v úvahu náhodné aj systematické zlyhania.
- Potreba demonštrovať robustnosť, nie len absenciu chýb.
- Aplikovať ohľad na bezpečnosť počas návrhu, výroby, servisu a likvidácie produktu.

SW inžinierstvo pre životne kritické systémy je mimoriadne náročné. Je vhodné stanoviť tri základné aspekty:

1. Procesy inžinierstva a manažmentu.
2. Výber vhodného nástroja a prostredia pre systém, ktoré umožní efektívne testovanie systému.
3. Určenie požiadaviek, ktoré spĺňajú regulačne a zákonom stanovené požiadavky.

Príklady životne kritických systémov

- *Infraštruktúra* – požiarny alarm, telekomunikácie, tiesňové služby.
- *Zdravotníctvo* – defibrilátor, chirurgické roboty, inzulínové a infuzné pumpy, srdcové stroje.
- *Jadrové inžinierstvo* – riadiaci systém jadrového reaktoru.
- *Rekreácia* – kolotoče.
- *Doprava* – riadiaci systém železničnej signalizácie.
- *Automobilový priemysel* – brzdný systém, airbag systém, riadiaci systém.
- *Letecký priemysel* – riadiace systémy vzdušnej premávky, avionické systémy, systémy podpory života posádky, riadiace systémy motoru, systémy plánovania letu.
- *Kozmický priemysel* – systémy v dopravných prostriedkoch určených pre let do vesmíru, systémy zabezpečujúce bezpečný štart rakety.

2.2 Bezpečnostné štandardy

SW štandardy môžeme rozdeliť do troch základných skupín, a to hodnotiace, vývojové a bezpečnostné [18].

Hodnotiace štandardy

Hodnotiace štandardy hodnotia proporcionalitu a kvalitu schopností organizácie a schopnosť tvorby SW. Medzi hodnotiace štandardy patrí napr. ISO 9000-3. Tieto štandardy pomáhajú identifikovať vývojárov s dobrými alebo zlými vývojovými procesmi. Medzi hodnotiace modeli patrí napr. Capability Maturity Model (CMM) [18].

Capabiliy Maturity Model stanovuje postup pre rozvoj zrelosti a schopnosti procesov. Poskytuje prostriedky pre zlepšenie kvality SW a výkonnosti vývoja. Kladie dôraz na plánovanie, školenia a podobne. Definuje päť úrovní zrelosti. Nezávisle hodnotí kľúčové oblasti procesov [9].

ISO 9000-3 je zameraný na zabezpečenie kvality. Je to dodatok k ISO 9000 zameraný na SW použitý pri vývoji, na testovanie, údržbu a produkciu SW [10].

Vývojové štandardy

Vývojové štandardy sú príručkou pri vytváraní reprodukovateľných vývojových SW procesov. Medzi vývojové štandardy patrí napr. DOD-STD-2167/2167A/2168/498, ISO/IEC 12207, DEF-STD 055/056. Tieto štandardy definujú dáta, ktoré majú byť vytvorené v priebehu vývojových procesov a podporujú údržbu týchto procesov.

DOD-STD-2167A/2168 štandardizuje vývojové procesy. Je flexibilný. Definuje tieto fázy: analýza systémových požiadaviek, návrh systému, analýza SW požiadaviek, predbežné preskúmanie návrhu, detailné preskúmanie návrhu, modulárne testovanie, testovanie integrácie komponent, testovanie konfigurácie SW, systémové integračné testovanie [11].

MIL-STD 498 je dobrovoľný, flexibilný štandard s cieľom nebrániť vývoju akýchkoľvek SW metód. Umožňuje záznam informácií v netradičných formátoch (napr. CASE nástroje) [12].

ISO/IEC 12207 stanovuje spoločný rámec pre životný cyklus SW, a to ako vyvíjať, dodávať, rozvíjať, prevádzkovať a udržiavať SW, ďalej ako tento rámec riadiť, napravovať a spravovať. Je procesne orientovaný. Je to základ v celosvetovom obchode so SW [13].

Bezpečnostné štandardy

Bezpečnostné štandardy poskytujú atribúty vývojových procesov SW, zodpovedajú otázky typu „Čo?“, ale nie „Ako?“. Tieto štandardy poskytujú väzby na bezpečnostné limity a poskytujú kritéria meraní. Táto kapitola sa zaoberá porovnaním požiadaviek na vývoj a certifikáciu životne kritického SW v rôznych oblastiach priemyslu. V tabuľke Tab.1 (zdroj [8]) je uvedené porovnanie niektorých bezpečnostných štandardov používaných v rôznych oblastiach priemyslu.

IEC 1508 je zameraný na bezpečnosť elektrických/elektronických/programovateľno-elektronických bezpečnostných systémov. Je používaný v celej Európe v bezpečnostných systémoch, v rôznych oblastiach (medicína, jadrové elektrárne a pod.). Je rozdelený do siedmich častí, pričom jedna je zameraná na SW [15].

Tab 1. Porovnanie vybraných bezpečnostných štandardov.

	Štandard				
Atribút	**IEC 1508**	**DO 178B**	**INT DEF 00-55**	**RIA 23**	**CENELEC**
Rozsah použiteľnosti	Všeobecný štandard	Avionické systémy	Obrana UK	Železničná signalizácia	Železničná signalizácia
Rozsah	Veľký	Veľký: pokrýva celý životný cyklus SW	Stredný: pokrýva návrh a V&V	Veľký: pokrýva celý životný cyklus SW	Veľký: pokrýva celý životný cyklus SW
Prijatie	Stredné: RIA 23 a CENELEC sú odvodené z tohto štandardu	Dobré: použitý pri rôznych civilných avionických projektoch	Chabé: malé množstvo projektov obrany UK	Stredné: použitý pri rôznych železničných projektoch po celom svete	Chabé: je aktuálne použitý po prvý krát
Klientela	Očakávajú sa odvodené štandardy, nemá konkrétny typ	Všetky veľké letecké spoločnosti	Niekedy používané ministerstvom obrany UK	Často používané spoločnosťou Bangladesh Railway a inými	Bude pravdepodobne používaný v rámci Európy
Budúcnosť	Dobrá	Dobrá	Chabá	Nahradzovaný štandardom CENELEC	Dobrá
Obtiažnosť na dodržiavanie	Stredná: je závislá na veľkosti prispôsobenia	Primeraná: požiadavky na testovanie sú ťažko dosiahnuteľné	Ťažká	Primeraná	Dosiaľ neurčená
Auditovateľ-nosť	Nízka	Stredná	Vysoká	Nízka	Stredná
Nezávislosť	Stredná	Nízka	Vysoká	Stredná	Vysoká

Keďže štandardy vychádzajú zo spoločného štandardu IEC 1508, zameriame sa práve na SW požiadavky definované v tomto štandarde:

Požiadavky na SW podľa IEC 1508

Nasledujúce a ďalšie informácie je možné nájsť v [15].

Špecifikácia bezpečnostných požiadaviek:

- Odvodená od špecifikovaných bezpečnostných požiadaviek na systém.
- Požiadavky musia byť dostatočne podrobné, aby umožnili návrh, implementáciu a funkčné posúdenie bezpečnosti.
- Požiadavky musia byť jasné, presné, overiteľné, testovateľné, udržiavateľné a uskutočniteľné.

- Požiadavky musia byť vhodné pre danú úroveň bezpečnosti a spätne trasovateľné k špecifikácii bezpečnostných požiadaviek systému.
- Terminológia musí byť jasná a zrozumiteľná.
- Musí obsahovať aj všetky potrebné požiadavky na seba-monitorovanie SW, diagnostické testy HW, periodické testovanie kritických funkcií.

Integrácia a testovanie:

- Testy integrácie medzi SW a HW sú vytvorené v priebehu návrhu a vývoja systému a špecifikujú:
    - Testovacie prípady a testovacie dáta v ovládaných integračných celkoch.
    - Testovacie prostredie, nástroje a konfigurácie.
    - Testovacie kritéria.
    - Procedúry pre nápravnú činnosť v prípade zlyhania testu.

Validácia SW bezpečnosti:

- Vykonáva sa ako celková kontrola, pre uistenie, že návrh SW spĺňa požiadavky na bezpečnosť SW.
- Môže byť vykonaná ako súčasť celkového hodnotenia systému
- Pre každú bezpečnostnú funkciu musia byť uvedené v dokumentácii záznam o overení činnosti, verzia SW, testovacie prostredie a výsledky.

Verifikácia SW:

- Testuje a hodnotí výsledky fázy životného cyklu - SW bezpečnosť pre uistenie, že sú v súlade so vstupnými informáciami.
- Overenie musí byť aplikované na požiadavky bezpečnosti SW, návrh SW architektúry, návrh architektúry systému, návrh SW modulov, zdrojový kód SW, dáta, modulárne testovanie SW, integračné testovanie SW, integračné testovanie HW, testovanie požiadaviek na bezpečnosť (validácia SW).

3 Vývoj a certifikácia softwaru v oblasti letectva

Táto kapitola popisuje procesy životného cyklu vývoja a certifikácie SW podľa štandardu RTCA/DO-178B [1].

3.1 Certifikačné organizácie

Vo svete existuje niekoľko organizácií zodpovedných za certifikáciu v oblasti letovej spôsobilosti. V Európskej únii je to **EASA** (European Aviation Safety Agency) so sídlom v Kolíne v Nemecku, je zodpovedná za certifikáciu v oblasti letovej spôsobilosti, a to všetkých leteckých výrobkov, častí a zariadení navrhnutých, vyrobených, udržovaných alebo používaných osobami v rámci regulačného dohľadu členských štátov EÚ. Certifikačná práca zahŕňa taktiež všetky post-certifikačné aktivity, ako je napríklad schvaľovanie zmien a opráv leteckej techniky a jej komponent, ako aj aktualizovanie a vydávanie letovej spôsobilosti k ochrane pred potenciálne nebezpečnými situáciami. Tieto a bližšie informácie k organizácii EASA sú uvedené v [6]. Medzi ďalšie organizácie patria napríklad FAA (Federal Aviation Administration s pôsobením v USA). Je dôležité si uvedomiť, že sa necertifikuje len software, ale certifikuje sa celý systém [3].

3.2 Úvod do RTCA/DO-178B

Prudký nárast aplikácie programového vybavenia v palubných systémoch a výstroji lietadiel a v motoroch z počiatku 80. rokov vyústil v potrebu nejakého návodného materiálu pre plnenie požiadaviek letovej spôsobilosti. Na uspokojenie tejto potreby bol vytvorený dokument DO-178B, Software Consideration in Airborne Systems and Equipment Certification (Programové vybavenie leteckých palubných systémov a výstroje z pohľadu osvedčovania jej spôsobilosti). Tento dokument bol vypracovaný komisiou RTCA a bol schválený v roku 1992. RTCA je asociácia leteckých organizácii Spojených štátov amerických ako štátnych, tak aj priemyselných [1].

DO-178B je o konzistencii a determinizme SW. Je potrebné demonštrovať trasovateľnosť zhora nadol na poukázanie, že požiadavky sú plne implementované. Je potrebné demonštrovať trasovateľnosť zdola nahor na poukázanie, že iba špecifikovaný SW bol implementovaný. DO-178B poskytuje okrem možnosti certifikácie aj ďalšie výhody, a to verifikáciu kvality SW, vysokú spoľahlivosť, konzistenciu, väčšiu možnosť znovupoužiteľnosti, nižšie náklady na životný cyklus, menšie náklady na údržbu, vyššiu rýchlosť integrácie HW a väčšiu pravdepodobnosť zachytenia chýb počas jednotlivých vývojových procesov [3].

Účelom tohto dokumentu je poskytnúť pokyny pre vznik programového vybavenia leteckých palubných systémov a výstroje, ktoré bude vykonávať potrebné funkcie s takou úrovňou istoty v otázke bezpečnosti, ktorá bude v súlade s požiadavkami letovej spôsobilosti. Tieto pokyny majú formu:

- Cieľov stanovených pre procesy životného cyklu softwaru.
- Popis činností a konštrukčných úvah vedúcich k dosiahnutí týchto cieľov.
- Popis dôkazov, ktoré potvrdia, že dané ciele boli dosiahnuté.

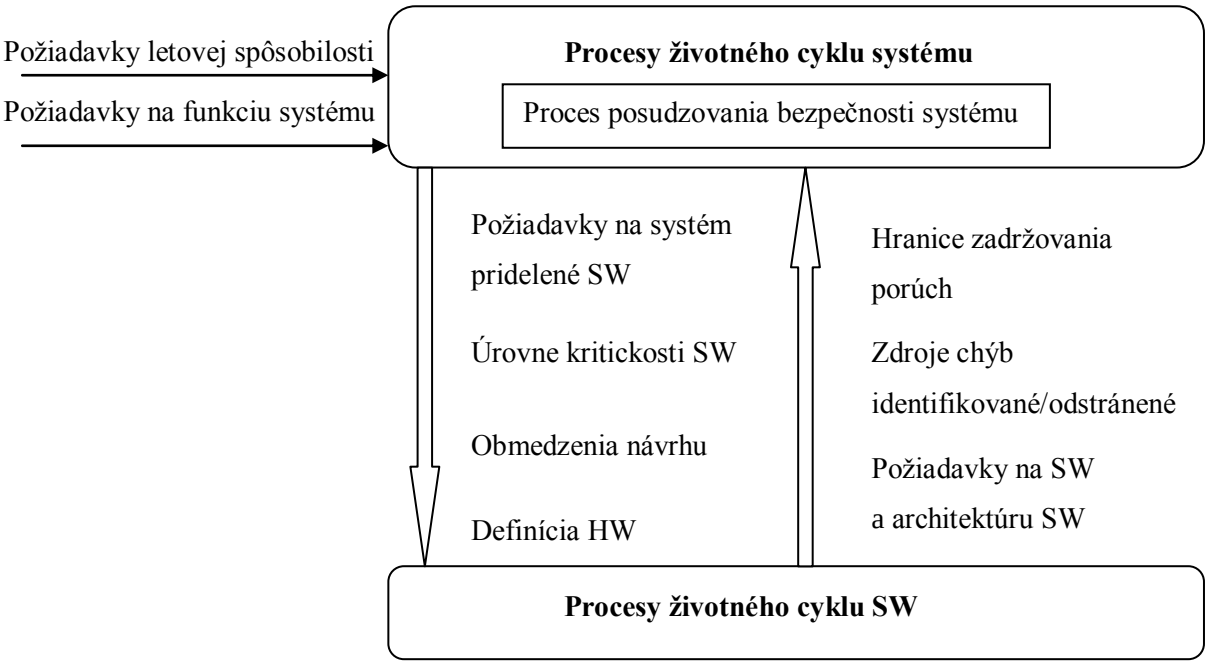
Tieto a ďalšie informácie ohľadom účelu a predmetu dokumentu DO-178B sú uvedené v [1].

3.3 Systémové aspekty

Táto kapitola popisuje aspekty procesov životného cyklu systému, ktoré sú nevyhnutné pre pochopenie procesov životného cyklu SW. Informácie tejto kapitoly sú čerpané z [1] a [2].

3.3.1 Informačný tok medzi procesmi životného cyklu systému a softwaru

Na Obr.1 je prehľad bezpečnostných aspektov informačného toku medzi procesmi životného cyklu systému a procesmi životného cyklu SW [1].



Obr.1: Tok informácií spojených s bezpečnosťou systému medzi procesmi životného cyklu systému a SW.

Informačný tok zo systémových procesov do SW procesov

Proces posudzovania bezpečnosti systému stanovuje a triedi do kategórií poruchové stavy systému. Bezpečnostné požiadavky sú časťou požiadaviek na systém, ktoré tvoria vstupy do procesov životného cyklu SW. Aby sa zaistilo, že bezpečnostné požiadavky sú správne realizované počas celého životného cyklu SW, tak odkazujú typicky na [1]:

- Popis systému a definíciu HW.
- Požiadavky pre osvedčovanie spôsobilosti, vrátane príslušných federálnych leteckých predpisov FAR (USA), spoločných leteckých predpisov JAR (Európa), poradných obežníkov AC (USA) atď.
- Požiadavky na systém pridelené SW, vrátane funkčných, výkonových a bezpečnostných požiadaviek.
- Úroveň (úrovne) kritickosti SW a údaje opodstatňujúce ich stanovenie, poruchové stavy, ich kategórie a dodatočné funkcie pridelené SW.
- Bezpečnostnú stratégiu a obmedzenia konštrukčného návrhu.
- Pokiaľ je daný systém súčasťou iného systému, potom aj bezpečnostné požiadavky a poruchové stavy tohto systému.

Informačný tok zo SW procesov do systémových procesov

Proces posudzovania bezpečnosti systému stanovuje dopad návrhu SW a jeho realizácie na bezpečnosť systému, na to využíva informácie z procesov životného cyklu SW. Medzi tieto informácie okrem iných patria:

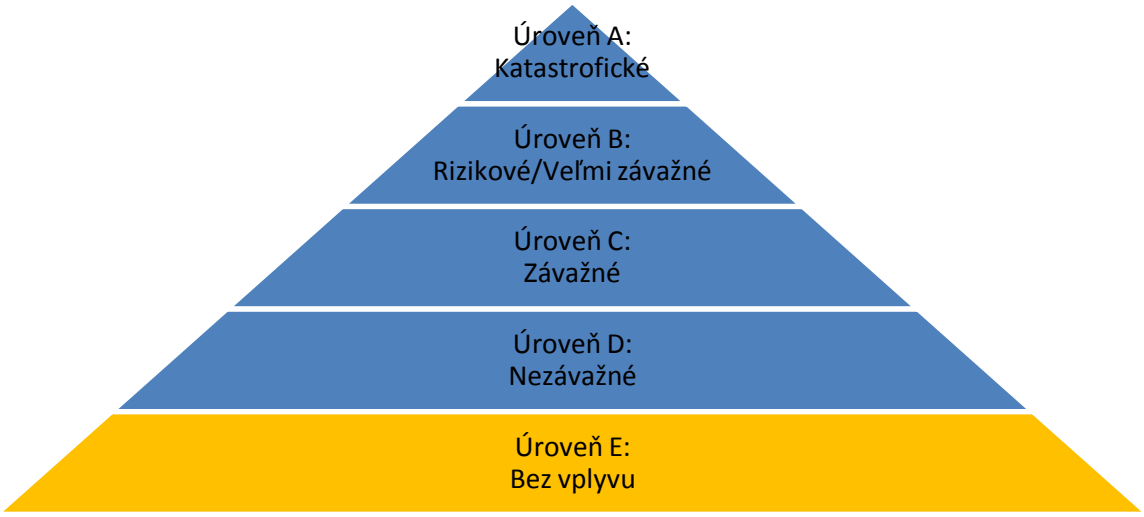
- Hranice zadržiavania porúch.
- Požiadavky na SW a architektúru SW.
- Zdroje chyb, ktoré boli identifikované alebo odstránené v celej architektúre SW použitím nástrojov alebo pomocou iných metód.

Trasovateľnosť medzi systémovými požiadavkami a údajmi konštrukčného návrhu SW je dôležitá pre proces posudzovania bezpečnosti SW. Modifikácie SW môžu ovplyvniť bezpečnosť systému, preto je potrebné ich identifikovať a predložiť procesu posudzovania bezpečnosti na vyhodnotenie [1].

3.3.2 Kategorizácia poruchových stavov

Poruchové stavy môžeme rozdeliť do piatich základných úrovní uvedených na obrázku Obr.2 [3], pričom platí:

- **Katastrofické:** Poruchové stavy, ktoré by zabránili v pokračovaní bezpečného letu a pristátia.
- **Rizikové/Veľmi závažné:** Poruchové stavy, ktoré by znížili spôsobilosť lietadla alebo schopnosť posádky zvládnuť nepriaznivé prevádzkové podmienky do takej miery, že by došlo k:
    - Značnému zníženiu bezpečnosti alebo funkčnej spôsobilosti.
    - Fyzickej úzkosti alebo k tak vysokému pracovnému zaťaženiu, že by sa nedalo spoliehať na posádku, že bude plniť svoje úlohy presne a úplne.
    - Nepriaznivým dopadom na posádku vrátane vážnych a potenciálne smrteľných zranení malého počtu týchto osôb.
- **Závažné:** Poruchové stavy, ktoré by znížili spôsobilosť lietadla alebo schopnosť posádky zvládnuť nepriaznivé prevádzkové podmienky do tej miery, kedy by došlo napríklad v výraznému zníženiu bezpečnosti alebo funkčnej spôsobilosti, miernemu zvýšeniu pracovnej záťaži posádky alebo k podmienkam zhoršujúcim výkonnosť posádky alebo vedúcim k značnému nepohodliu osadenstva, vrátane možných zranení.
- **Nezávažné:** Poruchové stavy, ktoré by výrazne neznížili bezpečnosť lietadla a znamenali by také činnosti posádky, ktoré sú úplne v rámci ich schopností. Nezávažné poruchové stavy zahrňujú napr. mierne zníženie bezpečnosti alebo funkčnej spôsobilosti, mierne zvýšenie pracovnej záťaže posádky, ako sú napríklad bežné zmeny v letovom pláne, alebo určité nepohodlie pre osadenstvo.
- **Bez vplyvu:** Poruchové stavy, ktoré neovplyvňujú prevádzkovú spôsobilosť lietadla.



Obr.2: Pyramída poruchových stavov.

3.3.3 Úrovne kritickosti softwaru

Úroveň kritickosti softwaru je založená na tom, ako software môže prispieť k potenciálnym poruchovým stavom. Úrovne kritickosti sú tieto [3]:

- **Úroveň A:** Software, ktorého anomálne chovanie spôsobí alebo prispeje k poruche, majúcej za následok pre lietadlo katastrofálny poruchový stav.
- **Úroveň B:** Software, ktorého anomálne chovanie spôsobí alebo prispeje k poruche, majúcej za následok pre lietadlo rizikový/veľmi závažný poruchový stav.
- **Úroveň C:** Software, ktorého anomálne chovanie spôsobí alebo prispeje k poruche, majúcej za následok pre lietadlo závažný poruchový stav.
- **Úroveň D:** Software, ktorého anomálne chovanie spôsobí alebo prispeje k poruche, majúcej za následok pre lietadlo nezávažný poruchový stav.
- **Úroveň E:** Software, ktorého anomálne chovanie nemá žiadny vplyv na prevádzkovú spôsobilosť lietadla alebo pracovné zaťaženie pilotov.

  V Tab.1 sú podrobnejšie porovnané úrovne kritickosti softwaru.

Tab.1: Porovnanie úrovni kritickosti.

Hľadisko podľa DO178B	**Úroveň A**	**Úroveň B**	**Úroveň C**	**Úroveň D**
Úroveň nezávislosti	Vysoká	Stredná	Nízka	Veľmi nízka
Nutnosť nízko úrovňových požiadaviek	Áno	Áno	Áno	Nie
Príkazové pokrytie kódu	Áno	Áno	Áno	Nie
Podmienkové pokrytie kódu	Áno	Áno	Nie	Nie
MCDC pokrytie kódu	Áno	Nie	Nie	Nie
Úroveň konfiguračného manažmentu	Vysoká	Vysoká	Stredná	Nízka
Korelačná analýza binárneho kódu	Áno	Nie	Nie	Nie
Verifikácia architektúry a algoritmov	Áno	Áno	Áno	Nie
Kontrola kódu	Áno	Áno	Áno	Nie

Vysvetlenie niektorých hľadísk z Tab.1:

- Úroveň nezávislosti – rozdelenie zodpovednosti, ktorá zaisťuje dosiahnutie objektívneho hodnotenia.
- Príkazové pokrytie kódu – každý príkaz kódu bol vykonaný aspoň raz.
- Podmienkové pokrytie kódu – každý bod vstupu a výstupu programu bol vyvolaný aspoň raz a každá podmienka nadobudla každý možný výsledok aspoň raz.
- MCDC pokrytie kódu - každý bod vstupu a výstupu programu bol vyvolaný aspoň raz, každá podmienka nadobudla každý možný výsledok aspoň raz a v každom rozhodnutí podmienky

bolo preukázané, že nezávisle ovplyvňuje výsledok podmienky. To znamená, že každý možný vstupný stav podmienky bol vykonaný aspoň raz.

- Konfiguračný manažment – proces identifikácie a definície konfigurácie systému, zaznamenávanie a podávanie správ o stave konfigurácie, tvorba požiadaviek na zmeny a overovanie úplnosti a správnosti konfigurácie.
- Korelačná analýza binárneho kódu – testovanie výsledného binárneho kódu a teda aj funkčnosti kompilátora.
- Kontrola kódu – kontrola, či zdrojový kód spĺňa požiadavky na formátovanie.

3.3.4 Stanovenie úrovne kritickosti softwaru

Na začiatku procesu posudzovania bezpečnosti systému stanovujeme softwarovú úroveň kritickosti, bez ohľadu na konštrukčný návrh systému. Pri stanovovaní úrovne kritickosti softwaru sa zvažuje dopad poruchy, ako pri strate funkcie tak aj pri chybnej funkcii.

Ak anomálne chovanie nejakej zložky SW prispieva k výskytu viacero poruchových stavov, potom sa stanovuje úroveň kritickosti na základe poruchového stavu najprísnejšej kategórie. Existujú rôzne stratégie výstavby architektúry systém, ktoré s postupom návrhu môžu vyústiť v prehodnotenie úrovne kritickosti SW, niektoré sú uvedené napríklad v [1].

Určitá funkcia systému môže byť pridelená jednej alebo viacerým izolovaným SW zložkám. Pri paralelnej implementácii je určitá funkcia systému realizovaná (implementovaná) viacnásobne, pomocou niekoľkých SW zložiek, a teda k výskytu poruchového stavu je potrebné anomálne chovanie viacerých SW zložiek. Pri paralelnej realizácii musí mať minimálne jedna SW zložka úroveň kritickosti najprísnejšej kategórie poruchových stavov pripadajúcich v úvahu. Úroveň kritickosti ostatných SW zložiek je stanovená podľa tej kategórie poruchových stavov, ktorá odpovedá strate funkcie.

Pri sériovej realizácii sú pre danú systémovú funkciu použité viaceré SW zložky tak, že anomálne chovanie ľubovoľnej zložky by mohlo vyvolať poruchový stav. Pri tejto realizácii budú mať SW zložky priradenú úroveň kritickosti odpovedajúcu najprísnejšej kategórii poruchových stavov stanovenej pre danú funkciu systému.

To, že software je vyvíjaný podľa nejakej úrovne kritickosti nenaznačuje nič o intenzite jeho porúch.

3.3.5 Požiadavky na systém vo vzťahu k overovaniu softwaru

Požiadavky na systém sú vytvorené z funkčných požiadaviek na systém a z bezpečnostných požiadaviek, ktoré sú výsledkom posudzovania bezpečnosti systému [1]. Je potrebné zahrnúť nasledujúce:

- Systémové požiadavky na letecký palubný software požadujú dve vlastnosti softwaru:

- Software vykonáva určené funkcie, definované požiadavky na systém.
- Software nevykazuje žiadne zvláštne anomálne správanie, čo je zistené procesom posudzovania bezpečnosti systému. V prípade výskytu anomálneho správania vznikajú dodatočné systémové požiadavky sledujúce jeho odstránenie.

- Tieto požiadavky na systém majú byť ďalej rozpracované do vysokoúrovňových požiadavkou na software, ktoré sú overované pomocou činností procesu overovania softwaru.

3.4 Procesy životného cyklu softwaru

Pokyny dokumentu DO-178B nepredpisujú žiadny prednostný životný cyklus softwaru, ale samostatne popisujú jednotlivé procesy, ktoré tvoria väčšinu životných cyklov a vzájomné pôsobenie medzi nimi.

Procesy životného cyklu softwaru sú:

- Proces plánovania softwaru, ktorý pre daný projekt vymedzuje a koordinuje činnosti vývojových a pridružených procesov softwaru.
- Vývojové procesy softwaru, ktorých výsledkom je softwarový produkt. Týmito procesmi sú:
  - Proces tvorby požiadaviek na software.
  - Proces návrhu softwaru.
  - Proces kódovania softwaru.
  - Proces integrácie softwaru.
- Pridružené procesy, ktoré zaisťujú pre procesy životného cyklu softwaru a ich výstupy správnosť, riadenie a dôveru. Tvoria najväčšiu položku nákladov projektu, tieto procesy pri certifikácií majú kľúčovú rolu. Pridruženými procesmi sú:
  - Proces verifikácie softwaru.
  - Proces riadenia konfigurácie softwaru.
  - Proces zabezpečovania kvality softwaru.
  - Proces styku s certifikačným orgánom.

Projekt definuje jeden alebo viacero životných cyklov SW tým, že pre každý proces volí činnosti, určuje ich poradie a prideľujem im zodpovednosti. U konkrétneho projektu je priradzovanie týchto procesov určené charakteristickými znakmi projektu ako sú napr. funkčnosť a zložitosť systému, rozmernosť a zložitosť softwaru, stabilita požiadavkou, využitie predošlým vývojom dosiahnutých výsledkov, vývojové stratégie a dostupnosť hardwaru. Obvyklé vývojové procesy softwaru nasledujú za sebou v poradí: tvorba požiadavkou, návrh, kódovanie, integrácia.

Procesy životného cyklu SW môžu byť vykonávané iteratívne. Časovanie a počet opakovaní sa mení v závislosti na dosiahnutom prírastku vo vývoji systémových funkcií, na zložitosti, na rozvoji požiadaviek, na dostupnosti HW, na spätnej väzbe k predchádzajúcim procesom a na ďalších charakteristických znakoch projektu.

Táto práca sa bude hlavne zaoberať procesom overovania softwaru a procesom styku s certifikačným orgánom, ktoré sú popísané nižšie. Bližší popis všetkých procesov je možné nájsť v [1, 2, 3].

3.5 Proces plánovania softwaru

Výsledkom tohto procesu sú plány a normované požiadavky, ktoré riadia vývojové a pridružené procesy SW. Účelom procesu plánovania SW je vymedziť také prostriedky pre jeho tvorbu, ktoré uspokojujú požiadavky na systém a poskytujú takú konfidenčnú úroveň (úroveň istoty), ktorá bude zodpovedať požiadavkám letovej spôsobilosti. Efektívne plánovanie je rozhodujúcim faktorom pri tvorbe softwaru [1].

Päť základných plánov, viď [3], tvoria:

- Plán aspektov certifikácie SW (PACS).
- Plán zabezpečenia kvality (PZK).
- Plán konfiguračného manažmentu (PKM).
- Plán vývoja SW (PVS).
- Verifikačný plán SW (VPS).

Tri základné štandardy, viď [3], tvoria:

- Štandardy pre: požiadavky (ŠPP), návrh (ŠPN), kódovanie (ŠPK).

Ciele procesu plánovania spočívajú v tom [1], že:

a) Sú vymedzené činnosti vývojových procesov a pridružených procesov životného cyklu, ktoré budú aplikované na požiadavky na systém a je určená úroveň kritickosti SW.
b) Je stanovený životný cyklus SW, vrátane vzájomných vzťahov medzi procesmi, ich radenie, spätne väzobné mechanizmy a prechodové kritéria.
c) Je zvolené prostredie životného cyklu SW, vrátane metód a nástrojov, ktoré majú byť použité pri činnostiach každého z procesov.
d) Pre vyvíjaný SW sú stanovené normy pre vývoj softwaru, ktoré odpovedajú cieľom bezpečnosti systému.
e) Sú vypracované SW plány.
f) Vývoj a revízie SW plánov sú koordinované.

3.6 Vývojové procesy softwaru

Vývojové procesy softwaru sa vykonávajú v súlade s tým, ako to stanovuje proces plánovania softwaru (viď kapitola 3.5) a PVS (bližší popis v [1]). Týmito procesmi sú:

- Proces tvorby požiadaviek na software.
- Proces návrhu softwaru.

- Proces kódovania softwaru.
- Proces integrácie softwaru.

Vývojové procesy softwaru vytvárajú jednu či viacero úrovní požiadaviek na software. **Vysokoúrovňové požiadavky VÚP** vznikajú priamo z analýzy systémových požiadaviek a architektúry systému. Obvykle sú tieto požiadavky ďalej rozpracované behom procesu návrhu softwaru, a tak vzniká jedna alebo viacero následných nižších úrovní požiadaviek na software. Avšak pokiaľ je zdrojový kód generovaný priamo z vysokoúrovňových požiadaviek, potom sú tieto požiadavky súčasne považované za požiadavky nízkoúrovňové a vzťahujú sa na nich príslušné pokyny pre požiadavky nízkoúrovňové.

Vývoj architektúry softwaru predstavuje radu rozhodnutí prijímaných ohľadom štruktúry softwaru. Behom procesu návrhu softwaru je definovaná architektúra a vznikajú požiadavky nízkoúrovňové. **Nízkoúrovňové požiadavky NÚP** sú požiadavky na software z ktorých je možné priamo vytvoriť zdrojový kód bez ďalších informácií.

Každý vývojový proces softwaru môže generovať odvodené požiadavky. **Odvodené požiadavky** sú požiadavky, ktoré nie sú priamo trasovateľné k požiadavkám vyššej úrovne. Príkladom vzniku takéhoto odvodeného požiadavku môže byť potreba vyvinúť pre zvolený cieľový počítač software pre spracovanie prerušení. Odvodené požiadavky môžu byť súčasťou ako požiadaviek vysokoúrovňových, tak požiadaviek nízkoúrovňových.

Proces tvorby požiadaviek na software

Ciele procesu tvorby požiadaviek na software sú:

a) Vytvorenie vysokoúrovňových požiadaviek, zahrňujú požiadavky na funkciu, výkonnosť, rozhranie a bezpečnosť.
b) Hlásenie odvodených vysokoúrovňových požiadaviek do procesu posudzovania bezpečnosti systému.

3.7 Proces verifikácie softwaru

Proces verifikácie softwaru patrí medzi pridružené procesy. Verifikácia je technické posudzovanie výsledkov vývojových procesov, ale aj procesov verifikácie softwaru. Proces verifikácie sa vykonáva v súlade s tým, ako to stanovuje proces plánovania softwaru (viď kapitola 3.5) a Popis plánu verifikácie softwaru (viď príloha 1).

Verifikácia nie je iba obyčajné skúšanie. Skúšanie neumožňuje dokázať neprítomnosť chýb. Táto kapitola sa zaoberá cieľmi a činnosťami procesu verifikácie SW, preskúmaniami a analýzami a procesom verifikácie SW [1].

3.7.1 Ciele procesu verifikácie softwaru

Účelom procesu verifikácie je odhaliť a hlásiť chyby, ktoré mohli byť do SW zavedené v priebehu vývojových procesov softwaru. Obecnými cieľmi procesu verifikácie SW je overiť, že:

- Z požiadaviek na systém, ktorých realizácia bola pridelená SW, sú odvodené vysokoúrovňové požiadavky na SW, ktoré požiadavky na systém pokrývajú.
- Z vysokoúrovňových požiadavkou je navrhnutá architektúra SW a odvodené nízkoúrovňové požiadavky, ktoré uspokojujú vysokoúrovňové požiadavky.
- Z architektúry softwaru a nízkoúrovňových požiadavkou je vytvorený zdrojový kód, ktorý uspokojuje nízkoúrovňové požiadavky a architektúru SW.
- Spustiteľný cieľový kód uspokojuje požiadavky na SW.
- Prostriedky použité k dosiahnutiu týchto cieľov sú technicky bezchybné a kompletné pre danú úroveň kritickosti SW.

3.7.2 Činnosti procesu verifikácie softwaru

Ciele procesu verifikácie softwaru sú dosiahnuté pomocou kombinácie preskúmania, analýz, vypracovaním jednotlivých skúšobných úloh a skúšobných postupov a postupným vykonaním týchto postupov. Preskúmanie a analýzy prinášajú posúdenie presnosti, kompletnosti a overiteľnosti požiadaviek na SW, architektúry SW a zdrojového kódu. Návrh skúšobných úloh môže priniesť ďalšie posúdenie konzistentnosti (vnútorného súladu) a kompletnosti požiadaviek. Vykonanie skúšobných postupov demonštruje splnenie požiadaviek.

Vstupy procesu verifikácie softwaru tvoria: požiadavky na systém, požiadavky na SW, architektúra SW, údaje o trasovateľnosti, zdrojový kód, spustiteľný cieľový kód a plán verifikácie softwaru.

Výstupy procesu verifikácie softwaru sú zaznamenané v dokumentoch: Úlohy a postupy overovania softwaru (viď príloha 1) a Výsledky procesu overovania softwaru (viď príloha 1).

Potreba overiteľnosti požiadaviek, ako náhle sú softwarovo realizované, môže sama o sebe znamenať uvalenie ďalších dodatočných požiadaviek alebo obmedzenie na vývojové procesy softwaru.

Verifikačný proces zaisťuje trasovateľnosť medzi realizáciou požiadaviek na SW a overením týchto požiadaviek:

- Trasovateľnosť medzi požiadavkami a skúšobnými úlohami je docielená pomocou analýzy pokrytia požiadaviek.
- Trasovateľnosť medzi kódovou štruktúrou a skúšobnými úlohami je docielená pomocou analýzy štrukturálneho pokrytia.

Návod pre činnosti procesu overovania softwaru zhrňuje toto:

a) Majú byť overené vysokoúrovňové požiadavky a trasovateľnosť k týmto požiadavkám.
b) Výsledky analýz trasovatelnosti a analýz pokrytia požiadaviek a štruktúry majú preukázať, že každá požiadavka na software je trasovateľná (vysledovateľná) až ku kódu, ktorý ju realizuje a až k preskúmaniu, analýze alebo skúšobnej úlohe, ktorá ju overuje.
c) Ak skúšaný kód nie je totožný so softwarom, majú byť rozdiely špecifikované a zdôvodnené.
d) Ak nie je možné určitú požiadavku na SW overiť skúšaním softwaru v reálnom skúšobnom prostredí, majú byť poskytnuté iné metódy, vrátane ich zdôvodnenia, ktoré povedú k splneniu cieľov procesu overovania softwaru a ktoré sú definované v doklade Plán verifikácie softwaru alebo v doklade Výsledky overovania softwaru.
e) Nedostatky a chyby objavené počas procesu overovania softwaru majú byť hlásené vývojovým procesom softwaru, aby sa vyjasnili a opravili.

3.7.3 Preskúmania a analýzy

Preskúmanie a analýzy sú aplikované na výsledky vývojových procesov SW a na proces verifikácie SW. Rozdiel medzi preskúmaniami a analýzami je ten, že analýzy poskytujú opakovane získateľne dôkazy správnosti a preskúmania poskytujú kvalitatívne hodnotenie správnosti. Preskúmanie sa môže skladať z kontroly výstupu procesu s pomocou kontrolného zoznamu alebo inej obdobnej pomôcky. Analýza môže detailne skúmať funkčnosť, výkon, trasovateľnosť a bezpečnostné dôsledky softwarovej zložky a ich vzťah k ostatným zložkám leteckého palubného systému alebo výstroji lietadla.

Preskúmanie a analýzy vysokoúrovňových požiadaviek

Cieľom týchto preskúmaní a analýz je odhaliť a hlásiť chyby v požiadavkách, ktoré tam mohli byť zavlečené v priebehu procesu tvorby požiadaviek. Tieto preskúmania a analýzy potvrdzujú, že vysokoúrovňové požiadavky splňujú tieto ciele:

a) **Súlad s požiadavkami na systém:** Cieľom je zaručiť, že sú vymedzené funkcie systému, ktoré budú vykonávané pomocou softwaru a že vysokoúrovňové požiadavky uspokoja požiadavky na funkčnosť, výkonnosť a bezpečnosť systému a že odvodené požiadavky a dôvody ich existencie sú správne stanovené.
b) **Presnosť a konzistencia:** Cieľom je zaručiť, že každá vysokoúrovňová požiadavka je presná, jednoznačná a dostatočne podrobná a že požiadavky nie sú vzájomne v konflikte.
c) **Zlučiteľnosť s cieľovým počítačom:** Cieľom je zaručiť, že medzi vysokoúrovňovými požiadavkami a hardware/softwarovými vlastnosťami (parametrami) cieľového počítača neexistujú konflikty, špeciálne v otázkach dôb odozvy systému a vstupných/výstupných obvodoch hardwaru.
d) **Overiteľnosť:** Cieľom je zaručiť, že každú vysokoúrovňovú požiadavku je možné overiť.

e) **Súlad s normami:** Cieľom je zaručiť, že normy pre tvorbu požiadaviek na SW boli v priebehu procesu tvorby požiadaviek dodržované, a že odchýlky od týchto noriem boli zdôvodnené.
f) **Trasovateľnosť:** Cieľom je zaručiť, že funkčné, výkonnostné a bezpečnostné požiadavky na systém, ktorých realizácia bola pridelená SW, boli rozpracované do VÚP.
g) **Algoritmy:** Cieľom je zaručiť presnosť a správne chovanie navrhovaných algoritmov.

Preskúmanie a analýzy nízkoúrovňových požiadaviek

Cieľom týchto preskúmaní a analýz je odhaliť a hlásiť chyby v požiadavkách, ktoré tam mohli byť zavlečené v priebehu procesu návrhu softwaru. Tieto preskúmania a analýzy potvrdzujú, že nízkoúrovňové požiadavky splňujú tieto ciele:

a) **Súlad s VÚP:** Cieľom je zaručiť, že nízkoúrovňové požiadavky uspokojujú vysokoúrovňové požiadavky a že odvodené požiadavky a dôvody ich existencie sú správne stanovené.
b) **Presnosť a konzistencia:** Cieľom je zaručiť, že každá nízkoúrovňová požiadavka je presná, jednoznačná a že požiadavky nie sú vzájomne v konflikte.
c) **Zlučiteľnosť s cieľovým počítačom:** Cieľom je zaručiť, že medzi požiadavkami na software a hardware/softwarovými vlastnosťami (parametrami) cieľového počítača neexistujú konflikty, špeciálne v prístupe k prevádzkovým prostriedkom (napr. zaťažovanie zbernice), v otázkach dôb odozvy systému a vstupných/výstupných obvodov hardwaru.
d) **Overiteľnosť:** Cieľom je zaručiť, že každú nízkoúrovňovú požiadavku je možné overiť.
e) **Súlad s normami:** Cieľom je zaručiť, že normy pre tvorbu požiadaviek na SW boli v priebehu procesu návrhu softwaru dodržované, a že odchýlky od týchto noriem boli zdôvodnené.
f) **Trasovateľnosť:** Cieľom je zaručiť, že vysokoúrovňové požiadavky a odvodené požiadavky boli rozpracované do nízkoúrovňových požiadaviek.
g) **Algoritmy:** Cieľom je zaručiť presnosť a správne chovanie navrhovaných algoritmov.

Preskúmanie a analýzy architektúry softwaru

Cieľom týchto preskúmaní a analýz je odhaliť a hlásiť chyby, ktoré mohli byť zavlečené v priebehu vývoja architektúry SW. Tieto preskúmania a analýzy potvrdzujú, že architektúra SW splňuje tieto ciele:

a) **Zlučiteľnosť s VÚP:** Cieľom je zaručiť, že architektúra softwaru nie je v rozpore s vysokoúrovňovými požiadavkami, špeciálne pri tých funkciách, ktoré zaisťujú integritu systému.
b) **Konzistencia:** Cieľom je zaručiť, že medzi jednotlivými zložkami architektúry softwaru existuje správny vzťah.

c) **Zlučiteľnosť s cieľovým počítačom:** Cieľom je zaručiť, že medzi architektúrou softwaru a hardware/softwarovými vlastnosťami (parametrami) cieľového počítača neexistujú konflikty, špeciálne v inicializácii, asynchrónnej prevádzke, synchronizácii a prerušeniach.
d) **Overiteľnosť:** Cieľom je zaručiť, že architektúru softwaru je možné overiť, napr. že neobsahuje žiadne neohraničené rekurzívne programy.
e) **Súlad s normami:** Cieľom je zaručiť, že Normy pre tvorbu požiadaviek na software boli v priebehu procesu návrhu softwaru dodržované, a že odchýlky od týchto noriem boli zdôvodnené, obzvlášť obmedzenia zložitosti a návrhovej konštrukcie, ktoré by nesplňovali ciele bezpečnosti systému.

Preskúmanie a analýzy zdrojového kódu

Cieľom je odhaliť a hlásiť chyby, ktoré mohli byť zavlečené do kódu v priebehu procesu kódovania. Tieto preskúmania a analýzy potvrdzujú, že výstupy procesu kódovania softwaru sú presné, úplné a je možné ich overiť. Prvoradou starosťou je správnosť kódu s ohľadom na softwarové požiadavky a architektúru softwaru a súlad s normami pre kódovanie SW. Tieto preskúmania a analýzy obvykle obmedzujú zdrojový kód, majú zahrňovať:

a) **Súlad s NÚP:** Cieľom je zaručiť, že zdrojový kód je presný a úplný z pohľadu nízkoúrovňových požiadaviek a že nerealizuje žiadnu funkciu, ktorá by nebola podchytená v dokumentácii.
b) **Súlad s architektúrou SW:** Cieľom je zaručiť, že zdrojový kód vyhovuje z hľadiska toku riadení definovaných v architektúre SW.
c) **Overiteľnosť:** Cieľom je zaručiť, že zdrojový kód neobsahuje príkazy a štruktúry, ktoré nie je možné overiť a že kód nemusí byť upravovaný aby mohol byť skúšaný.
d) **Súlad s normami:** Cieľom je zaručiť, že normy pre kódovanie softwaru boli v priebehu vývoja kódu dodržované, obzvlášť v otázkach obmedzenia zložitosti a kódových obmedzení, ktoré by vyhoveli cieľom bezpečnosti systému. Otázka zložitosti zahrňuje stupeň previazanosti jednotlivých zložiek softwaru, úrovne vnorenia u riadiacich štruktúr programu a zložitosť logických a číselných výrazov. Táto analýza taktiež zabezpečuje, že odchýlky od uvedených noriem sú zdôvodnené.
e) **Trasovateľnosť:** Cieľom je zaručiť, že NÚP sú obsiahnuté v zdrojovom kóde.
f) **Presnosť a konzistencia:** Cieľom je konštatovať správnosť a bezkonfliktnosť zdrojového kódu, vrátane použitia zásobníku, ošetrení pretečení a správnej voľby presnosti (počtu bitov) v aritmetike pevnej radovej čiarky, súperení prevádzkových prostriedkov (zdrojov), časovaní pre časovo najpriaznivejší cyklus, spracovaní mimoriadnych situácií, použití neinicializovaných premenných a konštánt, otázky nepoužívaných premenných a konštánt a poškodenia dát vplyvom konfliktu medzi úlohami alebo prerušeniami.

Preskúmanie a analýzy výstupu procesu integrácie

Cieľom je zaručiť, že výsledky procesu integrácie sú úplné a správne a to podrobným skúmaním údajov o spojení a zavedení a mapy pamäti, to zahŕňa:

a) Nesprávne hardwarové adresy.
b) Prekrývanie oblasti pamäti.
c) Chýbajúce softwarové zložky.

Preskúmanie a analýzy skúšobných úloh, postupov a výsledkov

Cieľom je zaručiť, že skúšanie kódu bolo navrhnuté a vykonané presne a úplne. Táto téma zahŕňa:

a) **Skúšobné úlohy:** Overovanie skúšobných úloh je popísané v kapitole 3.7.4.
b) **Skúšobné postupy:** Cieľom je overiť, že skúšobné úlohy boli presne rozpracované do skúšobných postupov a očakávaných výsledkov.
c) **Výsledky skúšok:** Cieľom je zaručiť, že výsledky skúšok sú správne a že rozpory medzi skutočnými a očakávanými výsledkami sú vysvetlené.

3.7.4 Proces skúšania softwaru

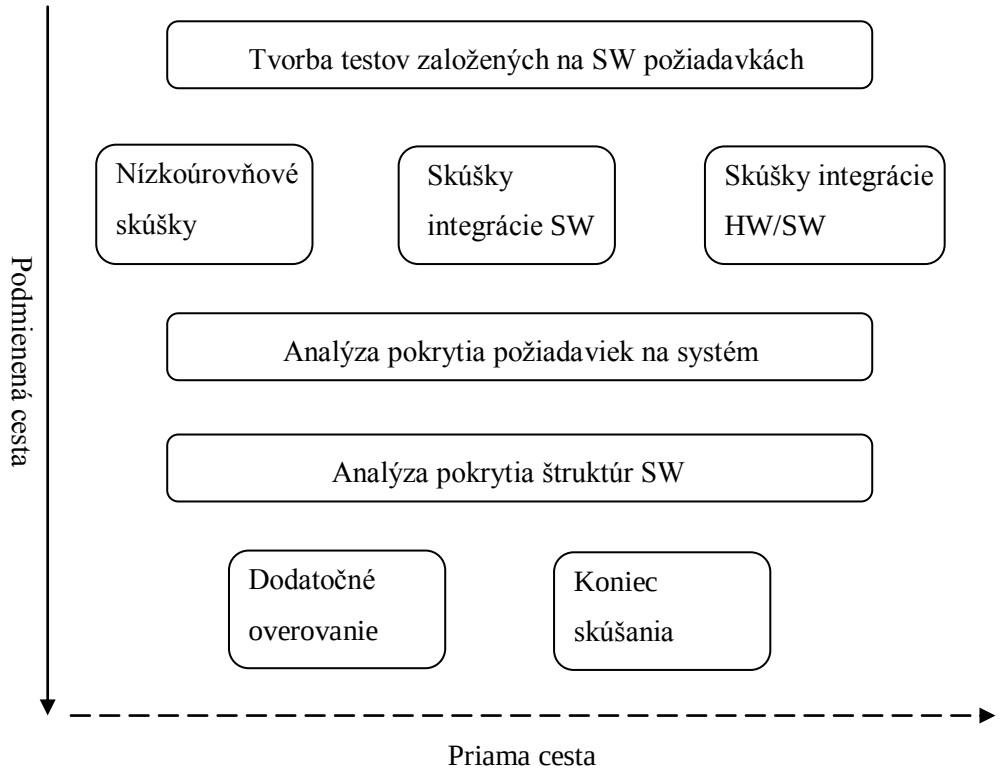
Skúšanie palubného softwaru má dva doplňujúce ciele. Jedným cieľom je dokázať, že software uspokojuje všetky požiadavky od neho očakávané. Druhým cieľom je dokázať, a to vysokou konfidenčnou úrovňou (stupňom istoty), že chyby, ktoré by mohli viesť k neprijateľným poruchovým stavom, ktoré definuje proces posudzovania bezpečnosti, sú odstránené.

Na Obr. 3 je diagram procesu skúšania softwaru. Ciele uvedených typov skúšania sú tieto:

- **Skúšanie integrácie hardware/software:** Overiť správnu funkciu softwaru v prostredí cieľového počítača.
- **Skúšanie integrácie softwaru:** Overiť vzájomné vzťahy medzi požiadavkami na software a softwarovými zložkami a overiť realizáciu softwarových požiadaviek a softwarových zložiek v rámci danej architektúry softwaru.
- **Nízkoúrovňové skúšanie:** Overiť realizáciu nízkoúrovňových požiadaviek na softwari.

Aby skúšanie softwaru dosiahlo svojich cieľov, platí že:

a) Skúšobné úlohy majú byť orientované predovšetkým na softwarové požiadavky.
b) Skúšobné úlohy majú byť vypracované tak, aby overili správnu funkčnosť a aby viedli ku stavom, pri ktorých sa odhalia potenciálne chyby.
c) Analýza pokrytia požiadaviek ma vymedziť, ktoré požiadavky na software neboli skúšané.
d) Analýza štrukturálneho pokrytia má vymedziť, ktoré softwarové štruktúry (štruktúry programu) neboli pri skúške preverované.



Obr. 3: Proces skúšania SW.

Skúšobné prostredie

Pre splnenie cieľov skúšania SW môže byť potrebné využiť viacero skúšobných prostredí. Väčšinou ideálne skúšobné prostredie prestavuje software zavedený priamo do cieľového zariadenia alebo skúšanie v prostrediu vysoko verne simulujúce prostredie cieľového počítača.

Certifikačnú dôveru je možné vložiť do skúšania vykonaného pomocou emulátoru cieľového počítača alebo pomocou simulátoru bežiaceho na hostiteľskom počítači. Návod týkajúci sa skúšobného prostredia zahrnuje, že vybrané skúšky majú byt vykonané v integrovanom prostredí cieľového počítača, pretože niektoré chyby je možné objaviť iba v tomto prostredí.

Výber požiadavkovo orientovaných skúšobných úloh

Na požiadavkovo orientované skúšanie je kladený dôraz, pretože bolo zistené, že táto stratégia je najefektívnejšia pri odhaľovaní chýb. Návod pre výber takýchto úloh zahrňuje:

a) Aby sa realizovali ciele skúšania SW, majú byť zapojene dve kategórie skúšobných úloh: skúšobné úlohy s hodnotami v normálnom rozsahu a skúšobné úlohy robustnosti (v abnormálnom rozsahu).
b) Určité skúšobné úlohy majú byť vypracované tak, aby vychádzali zo SW požiadaviek a zdrojov chýb zakotvených vo vývojových procesoch.

Skúšobné úlohy v normálnom rozsahu:

Cieľom skúšobných úloh v normálnom rozsahu je preukázať spôsobilosť softwaru reagovať na normálne vstupy a podmienky. Skúšobné úlohy normálneho rozsahu zahrňujú:

a) Vstupné premenné typu real a integer majú byť vyskúšané s použitím platných tried ekvivalencie a okrajových hodnôt.
b) U časovo závislých funkcií, ako sú filtre, integrátory a generátory oneskorení, má byť pri kontrole vlastností funkcie v danom kontexte použitých niekoľko iteračných cyklov.
c) Pre prechody medzi stavmi majú byť vypracované skúšobné úlohy tak, aby sa vyskúšali prechody, ku ktorým môže dôjsť v bežnej prevádzke.
d) Pri požiadavkách na software vyjadrených pomocou logických rovníc majú skúšobné úlohy v normálnom rozsahu preveriť používanie premenných a booleovské operátory.

Skúšobné úlohy robustnosti:

Cieľom skúšobných úloh robustnosti (odolnosti) je preukázať spôsobilosť softwaru reagovať na abnormálne vstupy a podmienky. Skúšobné úlohy na robustnosť zahrňujú toto:

a) Premenné typu real a integer majú byť preskúšané pomocou vybranej triedy ekvivalencie neplatných hodnôt.
b) Inicializácia systému má byť preskúšaná pri mimoriadnych podmienkach.
c) Majú byť stanovené druhy porúch vstupných dát, hlavne pri zložitých dátových reťazcoch prichádzajúcich z vonkajšieho systému.
d) Pri cykloch, kde počet prechodov cyklom je počítanou hodnotou, majú byť skúšobné úlohy vypracované tak, aby vypočítaná hodnota spadala mimo povolené medze a takto bola preukázaná robustnosť týchto časti kódu.
e) Má byť vykonaná kontrola, ktorá bude zaručovať, že ochranné mechanizmy, ktoré majú reagovať pri prekročení časovej lehoty, pracujú správne.
f) Pri časovo závislých funkciách, ako sú filtre, integrátory a generátory oneskorení, majú byť vypracované skúšobné úlohy zamerané na ochranné mechanizmy zasahujúce pri aritmetickom pretečení.
g) Pre skúšanie stavových prechodov majú byť vypracované skúšobné úlohy, ktoré budú vyvolávať prechody, ktoré nie sú softwarovými požiadavkami dovolené.

Požiadavkovo orientované metódy skúšania

Požiadavkovo orientované metódy skúšania sa skladajú z metód pre skúšanie integrácie hardware/software, skúšanie integrácie softwaru a nízkoúrovňového skúšania, ktoré vo všetkých prípadoch vychádzajú z požiadaviek. Popis týchto metód:

a) Požiadavkovo orientované skúšanie integrácie hardware/software: Táto metóda sa má zamerať na zdroje chýb spojených s fungovaním softwaru v prostredí cieľového počítača a na

funkcie z pohľadu vysokej úrovne. Cieľom skúšania tohto typu je poskytnuť záruku, že software bude v cieľovom počítači plniť vysokoúrovňové požiadavky. Typické chyby odhalené touto skúšobnou metódou sú:

- Nesprávne spracovania prerušení.
- Nesplnenie požiadaviek vymedzujúcich dobu vykonávania operácií.
- Nesprávna odozva softwaru na prechodové stavy hardwaru alebo poruchy hardwaru, napr. na nesprávne spúšťanie, na prechodné preťaženie vstupu atď.
- Problémy s dátovou zbernicou a s ďalšími prevádzkovými prostriedkami, napr. mapovanie pamäti.
- Neschopnosť vstavaného testu detekovať poruchy.
- Chyby v rozhraní hardware/software.
- Nesprávne chovanie spätno-väzobných slučiek.
- Nesprávne riadenie hardwaru pre správu pamäti alebo iných obvodov riadených softwarom.
- Pretečenie zásobníku.
- Nesprávna činnosť mechanizmu slúžiaceho k potvrdeniu správnosti a zlučiteľnosti zavádzaného softwaru.
- Porušenie hraníc izolujúcich funkčné bloky softwaru, ktoré vznikli pri rozklade.

b) Požiadavkovo orientované skúšanie integrácie softwaru: Cieľom skúšania integrácie softwaru založeného na požiadavkách je záruka, že softwarové zložky na seba správne vzájomne pôsobia a plnia požiadavky na software a sú v súlade s architektúrou softwaru. Táto metóda môže byť realizovaná postupným rozširovaním poľa pôsobnosti požiadaviek prostredníctvom postupného integrovania ďalších zložiek kódu spolu s príslušným rozširovaním rozsahu skúšobných úloh. Niektoré typické chyby odhalené touto metódou sú:

- Nesprávna inicializácia premenných a konštánt.
- Chyby pri predávaní parametrov.
- Poškodenie dát, zvlášť globálnych.
- Nedostatočný rozsah numerického rozlíšenia (end-to-end numerical resolution).
- Nesprávna postupnosť radenia udalostí a operácií.

c) Požiadavkovo orientované nízkoúrovňové skúšanie: Cieľom požiadavkovo orientovaného nízkoúrovňového skúšania je záruka, že softwarové zložky plnia nízkoúrovňové požiadavky. Typické chyby odhalené touto metódou zahrňujú:

- Zlyhanie algoritmu pri plnení jednej z požiadaviek na software.
- Nesprávne vykonania operácií cyklov.
- Nesprávne logické rozhodnutia.
- Zlyhanie pri správnom spracovaní povolených kombinácií vstupných stavov.

- Nesprávne odozvy na chýbajúce alebo poškodené dáta.
- Nesprávne spracovanie mimoriadnych situácií, ako sú aritmetické chyby a porušenie hraníc polí.
- Nesprávne poradie výpočtov.
- Nedostatočná presnosť alebo výkonnosť algoritmu.

Analýza pokrytia skúškami

Je to proces skladajúci sa z dvoch krokov, zahŕňa analýzu pokrytia založenú na požiadavkách a analýzu štrukturálneho pokrytia. V prvom kroku sa analyzujú skúšobné úlohy vo vzťahu k požiadavkám na software, aby sa potvrdilo, že splňujú vymedzené kritéria. Druhým krokom sa po potvrdzuje, že požiadavkovo orientované skúšobné postupy preverili kódovú štruktúru.

Analýza pokrytia požiadavkovo orientovanými skúškami

Jej cieľom je určiť, ako dobre požiadavkovo orientované skúšanie overilo realizáciu požiadavkov na software. Táto analýza ma preukázať že:

a) Pre každú požiadavku na software existujú skúšobné úlohy.
b) Skúšobné úlohy splňujú kritéria skúšania v normálnom rozsahu a skúšania robustnosti v súlade s tým, ako sú vymedzené v kapitole 3.7.4.

Analýza štrukturálneho pokrytia

Jej cieľom je určiť, na ktorú kódovú štruktúru neboli aplikované žiadne požiadavkovo orientované skúšobné postupy. Návod zahŕňa:

a) Analýza má potvrdiť stupeň štrukturálneho pokrytia stanovený primeranej úrovni kritickosti softwaru.
b) Analýza môže byť aplikovaná priamo na zdrojový kód za predpokladu, že SW nemá stanovenú úroveň kritickosti A a kompilátor negeneruje cieľový kód, ktorý nie je možné trasovať k príkazom zdrojového kódu. Ináč má byť cieľový kód dodatočne overený, aby sa potvrdila správnosť takto vygenerovaných postupností kódu.
c) Analýza má potvrdiť správnosť väzby v dátach a väzby v riadení medzi jednotlivými zložkami kódu.

Riešenie nedostatkov odhalených analýz štrukturálneho pokrytia

Analýza štrukturálneho pokrytia môže odhaliť kódovú štruktúru, ktorá behom skúšania nebola preverená. Táto nepreverená štruktúra môže byť výsledkom týchto faktorov:

a) Nedostatky v požiadavkovo orientovaných skúšobných úlohách alebo postupoch: Skúšobné úlohy majú byť doplnené alebo skúšobne postupy upravené tak, aby pokryli chýbajúce pokrytie.

b) Neprimeranosti v požiadavkách na software: Požiadavky na software majú byť upravené, majú byť vypracované a vykonané dodatočné skúšobné postupy.
c) Mŕtvy kód: Tento kód má byť odstránený a ma byť vykonaná analýza, ktorá odhadne vplyv tohto odstránenia a potrebu nového overovania.
d) Deaktivovaný kód: U deaktivovaného kódu, ktorý nie je určený k vykonaniu v žiadnej z konfigurácií použitých na lietadle alebo motore, majú kombinácie analýz a skúšok preukázať, že nemôže dôjsť k samovoľnému vykonaniu tohto kódu, je izolovaný. U deaktivovaného kódu, ktorý je vykonávaný iba v určitých konfiguráciách cieľového počítača, je potrebné nastaviť prevádzkovú konfiguráciu pre normálne vykonanie tohto kódu a vypracovať dodatočné skúšobné úlohy a postupy, aby sa dosiahlo požadovaných cieľov pokrytia.

3.8 Proces zabezpečovania kvality softwaru

Ciele procesu zabezpečovania kvality SW slúžia k získaniu dôvery, že procesy životného cyklu SW produkujú SW, ktorý odpovedá požiadavkám naňho kladeným a to tak, že zaistí, aby tieto procesy boli vykonané v súlade so schválenými plánmi a normami SW. Ciele procesu zabezpečovania kvality SW spočívajú v získaní záruk, že:

- Vývojové procesy SW a neoddeliteľne pridružné procesy vyhovujú vzhľadom k schváleným plánom a normám.
- Prechodové kritéria procesu životného cyklu SW sú splnené.
- Záverečné preskúmania kompletnosti a pripravenosti daného SW produktu sú vykonané.

3.9 Proces styku s certifikačným orgánom

Cieľom tohto procesu je určenie spôsobu komunikácie a spoločného prístupu medzi žiadateľom a certifikačným orgánom. Tento prístup má byť používaný po celý životný cyklus SW, aby sa tak podporil proces osvedčovania spôsobilosti (certifikačný proces).

Tento proces sa vykonáva v súlade s tým, ako to stanovuje proces plánovania SW (viď kapitola 3.5) a Plán softwarových aspektov osvedčovania spôsobilosti (viď príloha 1).

3.9.1 Spôsob plnenia požiadaviek a plánovania

Žiadateľ navrhuje spôsob plnenia požiadaviek, ktorý definuje ako vývoj leteckého palubného systému alebo výstroje bude plniť požiadavky predpisovej základni (certifikačnej bázy). Plán softwarových aspektov osvedčovania spôsobilosti definuje SW aspekty leteckého palubného systému alebo výstroje v kontexte navrhnutého spôsobu riešenia. Tento plán taktiež uvádza úroveň kritickosti SW tak, ako ju stanovil proces posudzovania bezpečnosti systému.

3.9.2 Dokladanie dôkazov o plnení

Žiadateľ poskytuje dôkazy o tom, že procesy životného cyklu SW plnia SW plány. Ich preskúmanie certifikačným orgánom sa môže konať u žiadateľa alebo u jeho dodávateľov. Žiadateľ sa dohoduje s certifikačným orgánom na týchto preskúmaniach a pripravuje doklady tak, aby boli dostupné podľa potreby.

3.9.3 Zoznam dokladov o životnom cykle softwaru

Minimálny zoznam dokladov o životnom cykle SW, ktoré sa predkladajú certifikačnému orgánu obsahuje:

- Plán softwarových aspektov osvedčovania spôsobilosti.
- Zoznam konfigurácie softwaru.
- Súhrn dosiahnutých výsledkov vývoja softwaru.

3.9.4 Údaje životného cyklu SW týkajúce sa typového návrhu

Pokiaľ certifikačný orgán nedovolí inak, predpisy týkajúce sa schvaľovania údajov životného cyklu SW, ktoré sa týkajú typového návrhu, sa vzťahujú na:

- Požiadavky na SW.
- Popis návrhu.
- Spustiteľný cieľový kód.
- Zoznam konfigurácie SW.
- Súhrn dosiahnutých výsledkov SW.

4 Návrh postupu verifikácie SW pre konkrétnu aplikáciu

V spolupráci s firmou UNIS a.s. bol vybraný SW, ktorý vykonáva riadenie elektronického zariadenia určeného pre riadenie chodu turbínového motoru. Toto zariadenie bude ďalej označované skratkou ZRM. Toto zariadenie je už zrealizované, rovnako aj daný SW, a v praxi sa požíva. Avšak v blízkej budúcnosti je tendencia ZRM certifikovať. Pri realizácii verifikačného plánu sa vychádzalo len z technickej špecifikácie, keďže žiadny zo základných plánov podľa DO-178B (viď kapitola 3.5) nebol doposiaľ vypracovaný.

Táto kapitola popisuje metódy a techniky pre validáciu a verifikáciu zariadenia ZRM. Avšak jedným z cieľov je tiež vytvoriť taký verifikačný plán, ktorý bude použiteľný aj pri iných projektoch s minimálnymi úpravami. Verifikovaný SW by mal spĺňať požiadavky na SW úrovne kritickosti „B“, v súlade so štandardom DO-178B. Úroveň kritickosti bola stanovená zákazníkom.

4.1 Teoretický pohľad na verifikačný plán

Na základe verifikačného plánu by sa malo postupovať v procese verifikácie softwaru, viď kapitola 3.7. Požiadavky na verifikačný plán alebo Plán verifikácie softwaru (viď príloha 1), sme už definovali. Tento plán by mal vychádzať taktiež z dokumentov Plán softwarových aspektov osvedčovania spôsobilosti, Plán zabezpečovania kvality softwaru, Plánu konfiguračného manažmentu. Napriek tomu, že tieto plány nie sú vypracované, je možné verifikačný plán vypracovať, pričom prípadné nezrovnalosti by sa v budúcnosti vyriešili.

Verifikačný plán verifikuje produkt použitím nástrojov, procesov a personálu. Rieši rozvrhnutie verifikácie, časový rámec a prideľovanie zdrojov. Plán musí byť adresovaný, t.j. pridelený niekomu, kto ho bude dodržiavať. Je typicky písaný tímom inžinierov, manažérom, prípadne externým expertom na danú problematiku. Manažment kvality zaručuje, že tento plán je prijatý a v súlade s DO-178B [3].

Je potrebné aby plán bol flexibilný. Napríklad vyberie sa daný testovací nástroj, ktorý bude monitorovať aj štrukturálne pokrytie kódu, avšak plán nebude špecifikovať konkrétne nastavenia nástroja, pretože užívateľské prostredie sa môže zmeniť [3].

Odporúčania pri vypracovávaní verifikačného plánu podľa [3]:

- Plán sa má hlavne zaoberať otázkou „Čo?“ a menej otázkou „Ako?“.
- Štandardy a kontrolné zoznamy nemajú byť súčasťou plánu, v pláne majú byť uvedené len referencie.
- Plán vypracovať pre danú spoločnosť tak, aby bol znovu použiteľný pri iných projektoch.

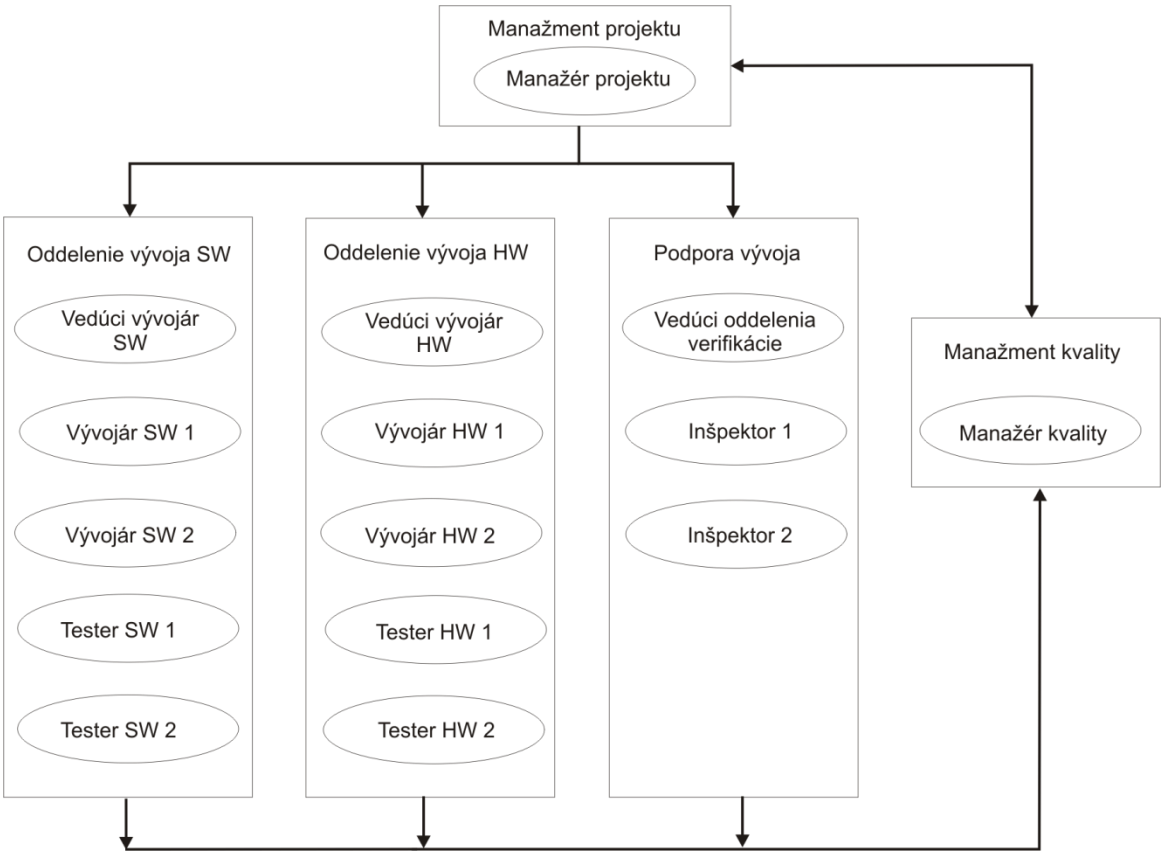
Kritéria hodnotenia verifikačného plánu podľa [3]:

- Je plán jasný a dostatočne detailný?
- Sú role členov verifikačného tímu vysvetlené?
- Sú zohľadnené aspekty nezávislosti?
- Sú popísané prístupy k požiadavkám, robustnosti a trasovateľnosti?
- Obsahuje popis použitých nástrojov a ich konfigurácií?
- Je schopná technická osoba, ktorej je princíp testovania a verifikácie neznámy, vykonávať testovanie iba na základe naštudovania informácií uvedených vo verifikačnom pláne?

4.2 Verifikačný plán

Jeho kostra je tvorená na základe dokumentu Popis plánu verifikácie softwaru (viď príloha 1). Je potrebné vypracovať úplne všetky body, aby sa predišlo pochybnostiam, zo strany certifikačnej organizácie, že sme na tieto body ignorovali, v našom prípade je to organizácia EASA.

4.2.1 Organizácia členov tímu



Obr. 4: Organizácia členov tímu.

Táto kapitola rozpracováva bod „a“, t.j. Organizácia, z Popis plánu verifikácie softwaru. Je potrebné si stanoviť organizáciu ľudí, ich počet, zodpovednosti a postavenie v organizácií. Model organizácie je zobrazený na Obr. 4. Základná delenie organizácie by malo byť nasledovné:

- *Manažment projektu* - zodpovedný za:
    - Zabezpečenie riadenia a koordinácie všetkých aktivít počas celého procesu vývoja.
    - Priebeh a výsledok projektu.
    - Kontrolu a schválenie *Plánu zabezpečenia kvality SW* a tiež je zodpovedný za kontrolu a schválenie všetkých dokumentov, ktoré sú výstupom procesu zabezpečenia kvality SW (kontrolné zoznamy a preskúmania).
    - Identifikáciu a implementáciu faktorov kvality do systému a SW.
    - Zabezpečenie riešenia nezhôd.
    - Vykonávanie pravidelných kontrol míľnikov projektu podľa harmonogramu.
- *Oddelenie vývoja SW* - vyvíjajú a vyrábajú produkty SW a SW podporu produktov. Toto oddelenie je zodpovedné za kompletizáciu technickej dokumentácie, testovanie, verifikáciu a zabezpečovanie kvality počas celého procesu vývoja SW. Zamestnanci vývoja SW sú rozdelený do dvoch skupín, jedna sa zaoberá vývojom a implementáciou SW, druhá implementáciou testovacích postupov a verifikáciou. Sú zodpovední za:
    - Implementáciu SW a realizáciu vývojových procesov a postupov podľa SW plánov a štandardov.
    - Riešenie otázok týkajúcich sa kvality návrhu a implementácie SW.
    - Identifikáciu, implementáciu a hodnotenie faktorov kvality, ktoré budú implementované v SW.
- *Oddelenie vývoja HW* – vyvíjajú a vyrábajú produkty HW a tiež HW podporu produktov (simulátory, pomocné testovacie zariadenia...). Toto oddelenie je zodpovedné za kompletizáciu technickej dokumentácie, testovanie, verifikáciu a zabezpečovanie kvality počas celého procesu vývoja HW. Rozdelenie zamestnancov vývoja HW a ich zodpovednosti sú podobné ako u zamestnancov vývoja SW, s rozdielom, že sa zaoberajú hardwarom.
- *Podpora vývoja* – zodpovedná za verifikáciu a za vývoj testovacích prípadov a postupov.
- *Manažment kvality* – stará sa o dodržiavanie systémových pravidiel, overovanie, rozvoj a zlepšovanie kvality projektu. Zabezpečuje dosiahnutie cieľov procesu zabezpečovania kvality SW (viď kapitola 3.8). Je potrebné aby toto oddelenie bolo nezávislé od projektu a vývojárov. Manažment kvality je zodpovedný za:
    - Vykonávanie vnútorného manažmentu kvality a vykonávanie interných auditov.
    - Kontrolovanie systému manažmentu kvality a všetkých dokumentov, ktoré s tým súvisia, počas celé procesu vývoja.
    - Identifikáciu a dokumentáciu nezhôd a rizík vzniknutých počas procesu vývoja.
    - Overenie opatrení a náprav vzniknutých nezhôd.

Poznámka

Ak sa zameriame len na SW, podľa [3] je minimálny počet ľudí organizácie štyri, pričom základné role sú: bezpečnostný expert, autor systémových požiadaviek, autor SW požiadaviek, autor HW požiadaviek, konfiguračný manažér, manažér kvality, SW architekt, vývojár, tester, manažér. V našom prípade je bezpečnostný expert zamestnancom externej firmy, ktorá taktiež stanovila úroveň kritickosti. Konfiguračný manažment je zabezpečený ostatnými členmi organizácie a pomocou vývojových nástrojov.

4.2.2 Nezávislosť

Táto kapitola rozpracováva bod „a“, t.j. Nezávislosť, z Popis plánu verifikácie softwaru. Pravidla nezávislosti boli stanovené na základe požiadaviek DO-178B pri úrovni kritickosti B. Tieto požiadavky sú nasledovné:

- Osoba ktorá overuje, či vysokoúrovňové požiadavky na SW vyhovujú požiadavkám na systém, by nemala byť identická s osobou, ktorá vysokoúrovňové požiadavky definovala.
- Osoba ktorá overuje, či vysokoúrovňové požiadavky na SW sú presné a konzistentné, by nemala byť identická s osobou, ktorá vysokoúrovňové požiadavky definovala.
- Osoba ktorá overuje, či nízkoúrovňové požiadavky na SW vyhovujú vysokoúrovňovým požiadavkám, by nemala byť identická s osobou, ktorá vysokoúrovňové požiadavky definovala.
- Osoba ktorá overuje, či nízkoúrovňové požiadavky na SW sú presné a konzistentné by nemala byť identická s osobou, ktorá nízkoúrovňové požiadavky definovala.
- Osoba ktorá vytvorila testovacie prípady a postupy by nemali byť identická s osobou, ktorá vytvorila nízkoúrovňové požiadavky.
- Osoba zodpovedná za vytvorenie testovacích postupov by nemala byť identická s osobou, ktorá testovacie postupy vytvorila.
- Osoba zodpovedná za vykonanie testovacích postupov a za to, že prípadné nezhody sú patrične vysvetlené, by nemala byť identická s osobou, ktorá bude daný test vykonávať.
- Osoba zodpovedná za vytvorenie výsledkov príkazového pokrytia kódu a za to, že prípadné nezhody sú patrične vysvetlené, by nemala byť identická s osobou, ktorá dané výsledky vytvorila.
- Osoba zodpovedná za vytvorenie výsledkov podmienkového pokrytia kódu a za to, že prípadné nezhody sú patrične vysvetlené, by nemala byť identická s osobou, ktorá dané výsledky vytvorila.
- Osoba zodpovedná za to, že vývojové a pridružené procesy vyhovujú schváleným plánom, by nemala byť identická s osobou, ktorá dané plány vytvorila.

- Osoba zodpovedná za to, že sú splnené prechodové kritéria pre procesy životného cyklu SW, by nemala byť identická s osobou, ktorá prechodové kritéria vytvorila.
- Osoba zodpovedná za to, že bolo vykonané preskúmanie kompletnosti a pripravenosti SW, by nemala byť identická s osobou, ktorá dané preskúmanie vykonala.

Tieto pravidlá a pravidlá pre ostatné úrovne kritickosti sú uvedené v [1].

4.2.3 Metódy preskúmania

Táto kapitola rozpracováva bod „c“, t.j. Overovacie metódy, z Popis plánu verifikácie softwaru.

Recenzia

Recenzovanie je proces, pri ktorom sa kontroluje vnútorná logika/štruktúra. Pri tomto procese sa taktiež vytvárajú aj kontrolné zoznamy. Proces recenzie je nasledovný:

1. Autor kontaktuje recenzenta a odovzdá mu dáta, ktoré je potrebné recenzovať.
2. V prípade potreby autor prediskutuje s recenzentom ciele produktu.
3. Recenzent, pomocou kontrolného zoznamu, vypracuje recenziu.
4. Recenzent prezentuje autorovi výsledky a prípadne návrhy na zlepšenie a nájdené chyby.
5. V prípade, že je potrebné vykonať korekcie, tak ich autor vykoná a celý proces recenzie sa zopakuje.

Inšpekcia

Inšpekcia na rozdiel od recenzie je viacej formálna, vyžaduje špecializáciu inšpektora. Proces inšpekcie je nasledovný:

1. Autor kontaktuje inšpektora a odovzdá mu zdrojový kód k inšpekcii.
2. Inšpektor vyhľadá a popíše prípadné chyby. Inšpekcia by sa mala vykonávať s rôznych ulov pohľadu, t.j. z pohľadu návrhu, kódovania, možnosti testovania a podobne.
3. V prípade potreby autor prediskutuje s inšpektorom nezrovnalosti.
4. Inšpektor prezentuje autorovi výsledky a prípadne návrhy na zlepšenie a nájdené chyby.
5. V prípade, že je potrebné vykonať korekcie, tak ich autor vykoná a celý proces inšpekcie sa zopakuje.

4.2.4 Analytické metódy

Táto kapitola rozpracováva bod „c“, t.j. Overovacie metódy, z Popis plánu verifikácie softwaru.

Vytváranie prototypu

Je to metóda, pri ktorej sa vytvárajú prototypy z dôvodu zistenia informácii o nejasných alebo vysoko rizikových aspektoch návrhu. Proces vytvárania prototypov je nasledovný:

1. Autor oboznámi testera s problematikou a cieľmi prototypu.
2. Tester vytvorí testovací postup, navrhne prostredie a spôsob tvorby prototypu.
3. Tester implementuje a optimalizuje algoritmus daného prototypu.
4. Tester algoritmus testuje a v prípade potreby upravuje prototyp.
5. Ak je daný prototyp vytvorený a otestovaný, tester vypracuje správu, ktorá zodpovie otázky, ktoré boli príčinou vzniku prototypu.
6. V prípade, že je potrebné vykonať korekcie zdrojového kódu, tak ich autor vykoná a celý proces vytvárania prototypu sa zopakuje.

Analýza trasovateľnosti

Všetky požiadavky sú zapísané v XML formáte, taktiež v tomto formáte sú aj zapísané všetky informácie o ich náväznosti. Následne sú z XML súborov pomocou XSLT jazyka vygenerované tabuľky trasovateľnosti a požiadavky v HTML formáte. Správnosť vygenerovaných tabuliek a následne analýza trasovateľnosti je overovaná manuálne. Proces analýzy trasovateľnosti je nasledovný:

1. Autor vytvorí matice trasovateľnosti.
2. Inšpektor skontroluje každú položku týchto matíc, vyhľadá a popíše prípadne chyby a nejasnosti.
3. Inšpektor prezentuje autorovi výsledky a prípadne návrhy na zlepšenie a nájdené chyby.
4. V prípade, že je potrebné vykonať korekcie, tak ich autor vykoná a celý proces analýzy sa zopakuje.

Analýza príkazového a podmienkového pokrytia kódu

Táto analýza sa vykonáva pomocou testovacieho SW nástroja Cantata++. Cieľom tejto analýzy je nájsť časti zdrojového kódu, ktoré neboli pokryté testovacími postupmi. V prípade nájdenia takéhoto kódu je potrebné buď daný kód odstrániť (ak sa jedná o tzv. „Mŕtvy kód“) alebo upraviť testovacie postupy (vytvoriť nový alebo rozšíriť aktuálne). V prípade že dôjde k vytvoreniu nového postupu, je možne, že bude potrebné dodatočne vytvoriť aj nové odvodené požiadavky. Proces analýzy príkazového a podmienkového pokrytia kódu je nasledovný:

1. Tester vykoná modulárne testovanie pomocou nástroja Cantata++, ktorá vygeneruje výsledok analýzy príkazového a podmienkového pokrytia testovaného kódu.

2. Tester spracuje výstup vytvorený nástrojom Cantata++ do výslednej správy, ktorú vyhodnotí.
3. V prípade nepokrytia nejakej časti testovaného kódu, tester modifikuje testovaciu procedúru a celý proces analýzy sa zopakuje.

Statická analýza kódu

Táto analýza sa vykonáva na zdrojovom kóde, teda bez nutnosti behu testovaného programu. Analýza je vykonávaná pomocou SW nástroja. Proces statickej analýzy kódu je nasledovný:

1. Autor kontaktuje inšpektora a odovzdá mu zdrojový kód k analýze.
2. Tester vykoná statickú analýzu pomocou nástroja Splint.
3. Tester spracuje výstup vytvorený nástrojom Splint do výslednej správy, ktorú vyhodnotí.
4. Tester prezentuje autorovi výsledky a prípadne návrhy na zlepšenie a nájdené chyby.
5. V prípade, že je potrebné vykonať korekcie, tak ich autor vykoná a celý proces analýzy sa zopakuje.

Analýza pokrytia požiadavkami

Cieľom tejto analýzy je nájsť požiadavky, pre ktoré neboli vytvorené testovacie postupy a následne vytvorenie týchto postupov. Proces analýzy pokrytia požiadavkami je nasledovný:

1. Autor kontaktuje inšpektora a odovzdá mu dáta k analýze.
2. Inšpektor overí, či sú všetky požiadavky pokryté testovacími prípadmi a či všetky testovacie prípady boli vykonané.
3. Inšpektor prezentuje autorovi výsledky a prípadne návrhy na zlepšenie a nájdené chyby.
4. V prípade, že je potrebné vykonať korekcie, tak ich autor vykoná a celý proces analýzy sa zopakuje.

Analýza pokrytia toku riadenia

Cieľom analýzy je zistiť kvalitu integračného testovania. Analýza zisťuje, ktoré toky riadenia neboli pomocou integračného testovania overené. V prípade nájdenia takéhoto toku riadenia, je potrebné vytvoriť nový testovací prípad, ktorý daný tok overí. Proces analýzy pokrytia toku riadenia je nasledovný:

1. Autor kontaktuje inšpektora a odovzdá mu dáta k analýze.
2. Inšpektor overí, či sú všetky toky riadenia v matici tokov riadenia pokryté testovacími prípadmi a či všetky testovacie prípady boli vykonané.
3. Inšpektor prezentuje autorovi výsledky a prípadne návrhy na zlepšenie a nájdené chyby.
4. V prípade, že je potrebné vykonať korekcie, tak ich autor vykoná a celý proces analýzy sa zopakuje.

4.2.5 Skúšobné metódy

Táto kapitola rozpracováva bod „c“, t.j. Overovacie metódy, z Popis plánu verifikácie softwaru. Základné princípy a súvislosti modulárneho, integračného a systémového testovania je možné nájsť v [4, 16, 17].

Modulárne testovanie

Tieto testy sú primárne založené na nízkoúrovňových požiadavkách. Modulárne testovanie zabezpečuje, že každá jedinečná časť projektu (modul) vykonáva svoju činnosť podľa presne zdokumentovanej špecifikácie a obsahuje jasne definované vstupy a očakávané výsledky. Modulárne testovanie predstavuje kombináciu troch druhov testovacích prístupov:

- **Biela skrinka:** jej cieľom je kontrola vnútornej logiky SW. Testy sú navrhnuté tak, aby overili vnútornú logiku modulu pomocou definovanej sady vstupných dát, ktoré testujú rôzne vnútorné logické vetvy. Každé takéto vstupné dáta sú samostatným testovacím prípadom.
- **Čierna skrinka:** jej cieľom je overiť funkčnosť SW. Pri testovaní sa pracuje s modulom ako s „čiernou skrinkou“, pričom vnútorný obsah je osobe vykonávajúcej testovanie neznámi. Tieto testy overujú špecifikáciu jednotlivých modulov.
- **Výkonnostné testy:** testujú, či SW spĺňa definované obmedzenia zdrojov. Napríklad modul musí vykonať danú funkciu v stanovenom čase alebo má spotrebovať menej ako stanovené množstvo pamäte.

Proces modulárneho testovania je nasledovný:

1. Autor kontaktuje testera a odovzdá mu zdrojový kód k testovaniu.
2. Tester implementuje testovacie simulačné prostredie a testovací skript na základe testovacieho postupu.
3. Tester vykoná modulárne testovanie na danom zdrojovom kóde.
4. Tester spracuje výstup testovania do výslednej správy, ktorú vyhodnotí.
5. Tester prezentuje autorovi výsledky a prípadne návrhy na zlepšenie a nájdené chyby.
6. V prípade, že je potrebné vykonať korekcie, tak ich autor vykoná a celý proces testovania sa zopakuje.

Systémové testovanie

Tieto testy sú primárne založené na vysokoúrovňových požiadavkách. Zabezpečujú, že celý SW systém spĺňa dané požiadavky. Testovanie systému sa skladá z týchto druhov testovacích postupov:

- **Funkčné testy:** Sú navrhnuté ako testy typu „čierna skrinka“ a overujú jednotlivé funkčné požiadavky.
- **Výkonnostné testy:** Overujú požiadavky na výkon, ktoré by mali stanovovať najhoršie prijateľné prípady, nominálne hodnoty a ideálne hodnoty ku ktorým je potreba smerovať.

Následne testy majú overiť, či boli vždy dosiahnuté prinajhoršom minimálne hodnoty, či sa zvyčajne dosiahli nominálne hodnoty a či boli niekedy dosiahnuté ideálne hodnoty.

- **Testy rozhrania:** Mali by byť navrhnuté tak, aby overili, že systém spĺňa požiadavky na externé rozhranie. Ak SW nebude môcť byť testovaný v prevádzkovom prostredí je nutné použiť simulátor alebo iné testovacie nástroje.
- **Operačné testy:** Zahŕňajú všetky testy užívateľského rozhrania, rozhrania človek stroj alebo človek počítač.
- **Testy zdrojov:** Požiadavky na využitie zdrojov, ako sú CPU, úložný priestor a veľkosti pamätí by mali byť stanovené vo vysokoúrovňových požiadavkách. Najlepším spôsobom ako otestovať splnenie týchto požiadaviek je prideliť systému len tieto zdroje a žiadne iné, ak dôjde k zlyhaniu zdroja, tak bol vyčerpaný. Prípadne je možné použiť alternatívne postupy ako napr. štatistických údajov o spotrebe zdrojov.
- **Bezpečnostné testy:** Bezpečnostné požiadavky musia stanovovať, že v prípade chyby sa SW musí snažiť predísť zraneniu osôb alebo poškodeniu majetku. Zhody s požiadavkami na bezpečnosť môžu byť testované nasledovné:
    - Pozorovaním správania sa systému pri poruchách v priebehu testovania.
    - Zámerným spôsobovaním porúch v kontrolovaných podmienkach a pozorovaním správania sa systému.

  Taktiež simulátory môžu byť navrhnuté na vykonávanie týchto testov.

Proces systémového testovania je nasledovný:

1. Autor kontaktuje testera a odovzdá mu spustiteľný kód k testovaniu.
2. Tester vytvorí testovacie prostredie a testovací skript na základe testovacieho postupu.
3. Tester vykoná systémové testovanie daného spustiteľného kódu.
4. Tester spracuje výstup testovania do výslednej správy, ktorú vyhodnotí.
5. Tester prezentuje autorovi výsledky a prípadne návrhy na zlepšenie a nájdené chyby.
6. V prípade, že je potrebné vykonať korekcie, tak ich autor vykoná a celý proces testovania sa zopakuje.

Integračné testovanie

Je to testovanie dvoch a viacerých modulov alebo funkcií súčasne s cieľom nájdenia chýb v ich rozhraní. Pri tomto testovaní je potrebné najprv zistiť všetky možné toky riadenia programu, z ktorých sa vytvorí strom toku riadenia programu [16] a matica ciest tohto stromu. Následne sa vytvoria integračné testy, proces integrácie môže byť zhora nadol alebo zhora nadol. Pri postupe zhora nadol je výhodou, že prípadná nutnosť opätovného návrhu je skôr odhalená. Pri postupe zdola nahor je výhodou, že moduly nižších úrovní, ktoré majú vyššiu pravdepodobnosť použitia sú otestované najskôr. V praxi sa zvyknú oba postupy kombinovať [17]. Proces integračného testovania je nasledovný:

1. Autor kontaktuje testera 1 a odovzdá mu zdrojový kód k testovaniu.
2. Tester 1 vytvorí maticu tokov riadenia a na jej základe aj testovacie postupy.
2. Tester 2 implementuje testovacie simulačné prostredia a testovacie skripty na základe testovacích postupov.
3. Tester 2 vykoná integračné testovanie na danom zdrojovom kóde.
4. Tester 2 spracuje výstup testovania do výslednej správy, ktorú vyhodnotí.
5. Tester 2 prezentuje autorovi výsledky a prípadne návrhy na zlepšenie a nájdené chyby.
6. V prípade, že je potrebné vykonať korekcie, tak ich autor vykoná a celý proces testovania sa zopakuje.

4.2.6 Verifikačné aktivity

Táto kapitola rozpracováva bod „c“, t.j. Overovacie metódy, z Popis plánu verifikácie softwaru. Verifikačné aktivity slúžia na pokrytie verifikačných požiadaviek. Verifikačné aktivity tvoria skúšobné (viď kapitola 4.2.5) a analytické metódy (viď kapitola 4.2.4) a metódy preskúmania (viď kapitola 4.2.3). Počas tohto procesu je potrebné verifikovať:

- SW plány a štandardy .
- Vysokoúrovňové požiadavky.
- Nízkoúrovňové požiadavky.
- SW architektúru.
- Zdrojový kód.
- Testovanie spustiteľného kódu.
- Výsledok verifikačného procesu.

Je potrebné u každého tohto bodu určiť metódu, aktivity, vstupy, výstupy a prostredie verifikácie.

Verifikácia SW plánov a štandardov

Aktivity: Základný formát – overujú sa formálne nezrovnalosti medzi SW plánmi a štandardami.

Korektnosť a úplnosť – overuje sa, či sú SW plány a štandardy presné, dôsledné, jasné a uskutočniteľné.

Zhoda s DO-178B – overujú sa nezrovnalosti SW plánov a štandardov s DO-178B, overuje sa pokrytie všetkých cieľov DO-178B.

Kompatibilita – overuje sa, že SW plány a štandardy sú kompatibilné v oblastiach popisu systému, životného cyklu, časového rozvrhu, zodpovedností, použitých metód a prostredí.

Metóda: Recenzia

Procedúra: Overovateľ recenzuje SW plány PVS, VPS, PZK, PKM, PACS a štandardy ŠPP, ŠPN, ŠPK v súlade s postupom popísaným v kapitole 4.2.3 Recenzie. Proces recenzovania bude kontrolovaný pomocou kontrolného zoznamu Kontrolný zoznam softwarových plánov a štandardov (viď príloha 2).

Vstupy: PACS, PZK, PKM, PVS, VPS, ŠPP, ŠPN, ŠPK, DO-178B, Kontrolný zoznam softwarových plánov a štandardov

Výstupy: Recenzie všetkých SW plánov a štandardov

Prostredie: MS Word

Verifikácia VÚP

Táto kapitola je vypracovaná na základe podkapitoly Preskúmanie a analýzy vysokoúrovňových požiadaviek z kapitoly 3.7.3.

Aktivity: Základný formát – overujú sa formálne nezrovnalosti medzi VÚP.

Presnosť a konzistentnosť – overuje sa, či sú VÚP presné, jednoznačné, dostatočne podrobné a že nie sú vzájomne v konflikte.

Súlad so systémovými požiadavkami – overuje sa, či sú VÚP kompatibilné s TS v oblastiach funkčných a výkonových charakteristík, požiadaviek na rozhranie, bezpečnosť, kvalitu, systémové zdroje.

Zlučiteľnosť s cieľovým počítačom – overuje sa, že VÚP a HW/SW vlastnosti cieľového počítača sú kompatibilné v oblastiach rozhraní, časovania a požiadaviek na zdroje.

Overiteľnosť – požiadavky majú byť vyjadrené z presného a kvantitatívneho hľadiska.

Súlad s ŠPP – overuje sa, že VÚP sú v súlade s ŠPP.

Trasovateľnosť k systémovým požiadavkám – overuje sa, že všetky VÚP sú trasovateľné k systémovým požiadavkám softwaru. Taktiež sa overuje, či sú všetky systémové požiadavky softwaru pokryté VÚP.

Metóda: Recenzia

Procedúra: Overovateľ recenzuje VÚP v súlade s postupom popísaným v kapitole 4.2.3 Recenzie. Proces recenzovania bude kontrolovaný pomocou kontrolného zoznamu Kontrolný zoznam softwarových požiadaviek (viď príloha 2).

Vstupy: VÚP, ŠPP, TS, Návrh HW, Matica trasovateľnosti, Kontrolný zoznam softwarových požiadaviek

Výstupy: Recenzia VÚP

Prostredie: MS Word

Aktivity: Presnosť algoritmov – najprv sa vytvorí prototyp a otestuje sa (v programe Matlab/Simulink), následne sa vykoná recenzia výsledku testu.

Metóda: Prototypovanie a recenzia

Procedúra: Prototyp sa vytvorí v súlade s postupom popísaným v kapitole 4.2.4 Prototypovanie. Overovateľ recenzuje VÚP v súlade s postupom popísaným v kapitole 4.2.3 Recenzie. Proces recenzovania bude kontrolovaný pomocou kontrolného zoznamu Kontrolný zoznam návrhu algoritmov (viď príloha 2).

Vstupy: VÚP, TS, Kontrolný zoznam návrhu algoritmov

Výstupy: Recenzie VÚP

Prostredie: MS Word, Matlab/Simulink

Verifikácia NÚP

Táto kapitola je vypracovaná na základe podkapitoly Preskúmanie a analýzy nízkoúrovňových požiadaviek z kapitoly 3.7.3.

Aktivity: Základný formát – overujú sa formálne nezrovnalosti medzi NÚP.

Presnosť a konzistentnosť – overuje sa, či sú NÚP presné, jedinečné, dostatočne podrobné a že nie sú vzájomne v konflikte.

Súlad s VÚP – overuje sa, či sú NÚP v súlade s VÚP v oblastiach značenia, funkčných a výkonových charakteristík, požiadaviek na rozhranie, bezpečnosť, kvalitu, systémové zdroje.

Zlučiteľnosť s cieľovým počítačom – overuje sa, že NÚP a HW/SW vlastnosti cieľového počítača sú kompatibilné v oblastiach rozhraní, časovania a požiadaviek na zdroje.

Overiteľnosť – požiadavky majú byť vyjadrené z presného a kvantitatívneho hľadiska.

Súlad s ŠPP – overuje sa, že NÚP sú v súlade s ŠPP.

Trasovateľnosť k VÚP – overuje sa, že všetky NÚP sú trasovateľné k VÚP. Taktiež sa overuje, či sú všetky VÚP pokryté NÚP.

Metóda: Recenzia

Procedúra: Overovateľ recenzuje NÚP v súlade s postupom popísaným v kapitole 4.2.3 Recenzie. Proces recenzovania bude kontrolovaný pomocou kontrolného zoznamu Kontrolný zoznam softwarových požiadaviek (viď príloha 2).

Vstupy: VÚP, ŠPP, TS, Návrh HW, Matica trasovateľnosti, Kontrolný zoznam softwarových požiadaviek

Výstupy: Recenzie NÚP

Prostredie: MS Word

Aktivity: Presnosť algoritmov – overujú sa implementačné veci ako presnosť zaokrúhľovania, delenie nulou, pretečenie a podtečenie, presnosť reprezentácie reálnych čísel a pod.

Metóda: Recenzia

Procedúra: Overovateľ recenzuje NÚP v súlade s postupom popísaným v kapitole 4.2.3 Recenzie. Proces recenzovania bude kontrolovaný pomocou kontrolného zoznamu Kontrolný zoznam algoritmov (viď príloha 2).

Vstupy: VÚP, TS, Kontrolný zoznam algoritmov

Výstupy: Recenzie NÚP

Prostredie: MS Word

Verifikácia architektúry SW

Táto kapitola je vypracovaná na základe podkapitoly Preskúmanie a analýzy architektúry SW z kapitoly 3.7.3.

Aktivity: Základný formát – overujú sa formálne nezrovnalosti v architektúre SW.

Konzistentnosť – overuje sa, či navrhnutá architektúra je optimálna pre navrhnutý software. Kontroluje sa, či je popis architektúry jasný, úplný a dostatočne detailný.

Súlad s VÚP – overuje sa, či architektúra SW je v súlade s VÚP v oblastiach integrity systému, funkčných a výkonových charakteristík, požiadaviek na rozhranie, bezpečnosť, kvalitu, systémové zdroje.

Zlučiteľnosť s cieľovým počítačom – overuje sa, že architektúra SW a HW/SW vlastnosti cieľového počítača sú kompatibilné v oblastiach rozhraní, časovania a požiadaviek na zdroje.

Overiteľnosť – požiadavky na architektúru SW majú byť vyjadrené z presného a kvantitatívneho hľadiska.

Súlad s ŠPP – overuje sa, že požiadavky na architektúru SW sú v súlade s ŠPP.

Metóda: Recenzia

Procedúra: Overovateľ recenzuje Architektúru SW v súlade s postupom popísaným v kapitole 4.2.3 Recenzie. Proces recenzovania bude kontrolovaný pomocou kontrolného zoznamu Kontrolný zoznam architektúry softwaru (viď príloha 2).

Vstupy: Architektúra SW, VÚP, Popis návrhu softwaru, Návrh HW, Kontrolný zoznam architektúry softwaru

Výstupy: Recenzie Architektúry SW

Prostredie: MS Word

Verifikácia zdrojového kódu

Táto kapitola je vypracovaná na základe podkapitoly Preskúmanie a analýzy zdrojového kódu z kapitoly 3.7.3.

Aktivity: Súlad s NÚP – overuje sa, či je zdrojový kód v súlade s NÚP, či sú implementované iba zdokumentované funkcie a všetky NÚP.

Súlad s architektúrou SW – overuje sa, že zdrojový kód vyhovuje z hľadiska toku riadená a dát definovaných v architektúre SW.

Overiteľnosť – overuje sa dostupnosť zdrojového kódu, použiteľnosť skúšobných nástrojov a testovacích metód.

Súlad s ŠPK – overuje sa, že zdrojový kód je v súlade s ŠPK.

Trasovateľnosť k NÚP – overuje sa, že zdrojový kód je trasovateľný k NÚP. Taktiež sa overuje, či sú všetky NÚP implementované.

Presnosť a konzistentnosť – overuje sa, či je zdrojový kód úplný a konzistentný, či sú funkčné rozhrania správne nadefinované, či formálne parametre a ich dátové typy sú zhodné s implementovanými parametrami a ich dátovými typmi.

Metóda: Recenzia

Procedúra: Overovateľ recenzuje zdrojový kód v súlade s postupom popísaným v kapitole 4.2.3 Recenzie. Proces recenzovania bude kontrolovaný pomocou kontrolného zoznamu Kontrolný zoznam recenzie zdrojového kódu (viď príloha 2).

Vstupy: Zdrojový kód, NÚP, Tabuľky trasovateľnosti, ŠPK, Kontrolný zoznam recenzie zdrojového kódu

Výstupy: Recenzie zdrojového kódu

Prostredie: MS Word

Aktivity: Inšpekcia zdrojového kódu – podrobne a metodicky sa vykonáva preskúmanie zdrojového kódu s cieľom nájsť výpočtové chyby, chyby referencie a deklarácie dát, chyby porovnania, chyby kontroly toku dát a riadenia a pod.

Metóda: Inšpekcia, statická analýza kódu

Procedúra: Overovateľ vykonáva statickú analýza zdrojového kódu v súlade s postupom popísaným v kapitole 4.2.4 Statická analýza kódu. Overovateľ vykonáva inšpekciu zdrojového kódu v súlade s postupom popísaným v kapitole 4.2.3 Inšpekcia. Proces recenzovania bude kontrolovaný pomocou kontrolného zoznamu Kontrolný zoznam inšpekcie zdrojového kódu (viď príloha 2).

Vstupy: Zdrojový kód, ŠPK, Kontrolný zoznam inšpekcie zdrojového kódu

Výstupy: Recenzie zdrojového kódu

Prostredie: MS Word, Splint

Testovanie spustiteľného kódu

Táto kapitola je vypracovaná na základe podkapitoly Preskúmanie a analýzy procesu integrácie z kapitoly 3.7.3.

Aktivity: Súlad s VÚP – je vykonávané systémové testovanie, pričom vstupom do testovania je úspešne integrovaný systém. Systém je testovaný v normálnych rozsahoch, to napríklad znamená, že:

- So vstupnými premennými sa pracuje v platných rozsahoch, t.j. neprekračujú sa hraničné hodnoty.
- U časovo závislých funkcií, ako sú filtre, integračné funkcie, funkcie oneskorenia, iteračné funkcie sú testované ich vlastnosti v rámci normálneho rozsahu.
- Testovacie procedúry, pre testovanie prechodu medzi jednotlivými stavmi systému, testujú tieto prechody pri bežných prevádzkových hodnotách.

Robustnosť s VÚP – je vykonávané systémové testovanie, pričom vstupom do testovania je úspešne integrovaný systém. Systém je testovaný v abnormálnych rozsahoch, to napríklad znamená, že:

- So vstupnými premennými sa pracuje v neplatných rozsahoch.
- Inicializácia systému je vykonávaná za abnormálnych podmienok.
- Systém je testovaný pri rôznych chybných vstupných dátach z externých systémov.
- U časovo závislých funkcií, ako sú filtre, integračné funkcie, funkcie oneskorenia, iteračné funkcie sú testované ich mechanizmy ochrany pri aritmetickom pretečení.
- Pre testovanie prechodu medzi jednotlivými stavmi systému sú vytvorené testovacie procedúry, ktoré sa pokúšajú vykonať prechod pri nepovolených situáciách.

Metóda: Systémové testovanie

Procedúra: Tester vykonáva systémové testovanie podľa postupu popísaného v kapitole 4.2.5 Systémové testovanie.

Vstupy: Spustiteľný kód, VÚP, Testovacie plány a prípady

Výstupy: Výsledky testovania SW

Prostredie: MS Word, TPZRM

Aktivity: Súlad s NÚP – testovanie na báze modulárneho testovania v normálnych rozsahoch, princípy testovania sú rovnaké ako u testovania súladu s VÚP.
Robustnosť s NÚP – testovanie na báze modulárneho testovania v abnormálnych rozsahoch, princípy testovania sú rovnaké ako u testovania robustnosti s VÚP.

Metóda: Modulárne testovanie

Procedúra: Tester vykonáva modulárne testovanie podľa postupu popísaného v kapitole 4.2.5 Modulárne testovanie.

Vstupy: Spustiteľný kód, NÚP, Testovacie plány a prípady

Výstupy: Výsledky testovania SW

Prostredie: MS Word, Cantata++

Aktivity: Súlad s architektúrou SW – overuje sa pomocou integračného testovania. Cieľom je zistiť, či interakcia komponent SW je správna a v súlade s požiadavkami na SW a jeho architektúru.

Metóda: Integračné testovanie

Procedúra: Tester vykonáva integračné testovanie podľa postupu popísaného v kapitole 4.2.5 Integračné testovanie.

Vstupy: Spustiteľný kód, Návrh SW

Výstupy: Výsledky testovania SW

Prostredie: MS Word

Aktivity: Kompatibilita s cieľovým počítačom – je overovaná pomocou analýzy výstupných súborov linkeru. Analyzuje sa, či je spustiteľný kód zostavený pre cieľový HW, či umiestnenie segmentov v pamäti je kompatibilné s cieľovým HW, či veľkosti segmentov zodpovedajú veľkostiam segmentov fyzickej pamäte.

Metóda: Recenzia

Procedúra: Overovateľ recenzuje výstup linkeru v súlade s postupom popísaným v kapitole 4.2.3 Recenzie. Proces recenzovania bude kontrolovaný pomocou kontrolného zoznamu Kontrolný zoznam kompatibility softwaru s cieľovým hardwarom (viď príloha 2).

Vstupy: Spustiteľný kód, Návrh HW, Dáta linkeru (mapa pamäti, tabuľka symbolov a pod.), Kontrolný zoznam kompatibility softwaru s cieľovým hardwarom

Výstupy: Výsledky testovania SW

Prostredie: MS Word

Verifikácia výsledkov procesov verifikácie

Táto kapitola je vypracovaná na základe podkapitoly Preskúmanie a analýzy skúšobných úloh, postupov a výsledkov z kapitoly 3.7.3.

Aktivity: Správnosť testovacích procedúr – overuje sa, či testovacie procedúry zodpovedajú požiadavkám a formálna správnosť verifikácie.
Správnosť testovacích výsledkov a vysvetlení rozporov – overujú sa výsledky testovania, taktiež sa overuje, či výsledky všetkých testov boli úspešné, prípadné nezrovnalosti boli zaznamenané a opravy vykonané.

Metóda: Recenzia

Procedúra: Overovateľ recenzuje výsledky testovania v súlade s postupom popísaným v kapitole 4.2.3 Recenzie. Proces recenzovania bude kontrolovaný pomocou kontrolného zoznamu Kontrolný zoznam výsledkov testovania (viď príloha 2).

Vstupy: VÚP, NÚP, Výsledky testovania, Kontrolný zoznam výsledkov testovania

Výstupy: Verifikované výsledky testovania SW

Prostredie: MS Word

Aktivity: Dosiahnutie pokrytia VÚP testami – overenie, že všetky VÚP boli pokryté testami a že všetky testy boli vykonané.
Dosiahnutie pokrytia NÚP testami – overenie, že všetky NÚP boli pokryté testami a že všetky testy boli vykonané.

Metóda: Recenzia

Procedúra: Overovateľ vykonáva analýzu pokrytia skúškami v súlade s postupom popísaným v kapitole 4.2.4 Analýza pokrytia skúškami.

Vstupy: VÚP, NÚP, Testovacie postupy

Výstupy: Správa o analýze pokrytia požiadaviek

Prostredie: MS Word

Aktivity: Dosiahnutie príkazového a podmienkového pokrytia kódu – overenie dosiahnutia tohto pokrytia je vykonávané pomocou nástroja Cantata++. Testovacími postupmi musí byť úplne (príkazovo aj podmienkovo) pokrytý zdrojový kód.

Metóda: Recenzia

Procedúra: Overovateľ vykonáva analýzu príkazového a podmienkového pokrytia zdrojového kódu v súlade s postupom popísaným v kapitole 4.2.4 Analýza príkazového a podmienkového pokrytia zdrojového kódu.

Vstupy: Zdrojový kód, Dáta o štrukturálnom pokrytí (vytvára Cantata++)

Výstupy: Správa o analýze príkazového pokrytia kódu, Správa o analýze podmienkového pokrytia kódu

Prostredie: MS Word, Cantata++

Aktivity: Dosiahnutie pokrytia toku riadenia – overenie, že celý strom toku riadenia bol pokrytý integračnými testami.

Metóda: Recenzia

Procedúra: Overovateľ vykonáva analýzu pokrytia toku riadenia v súlade s postupom popísaným v kapitole 4.2.4 Analýza pokrytia toku riadenia.

Vstupy: Výsledky integračného testovania, testovacie procedúry, strom toku riadenia

Výstupy: Správa o analýze pokrytia toku riadenia

Prostredie: MS Word

4.2.7 Overovacie prostredie

Táto kapitola rozpracováva bod „d“, t.j. Overovacie prostredie, z Popis plánu verifikácie softwaru.
Pri všetkých verifikáciách a testovacích činnostiach na ZRM je použitá platforma PC s operačným systémom Windows XP. Ďalej bude platiť:

- **Verifikácia požiadaviek -** Požiadavky sú vyvíjané použitím XML technológie a následne transformované do HTML formátu. Finálne sú verifikované manuálne. Analýza trasovateľnosti je čiastočne splnená počas procesu transformácie XML do HTML.
- **Prostredie pre tvorbu recenzií –** Sú použité nástroje MS Word a email. MS Word je použitý pre vypracovanie správy o kontrole a email pre komunikáciu.
- **Statická analýza zdrojového kódu –** Sú vykonávané pomocou nástroja Splint. Splint je nástroj pre statickú kontrolu programov v jazyku C, ktorá pomáha pri odhaľovaní zraniteľnosti kódu a programovacích chýb [14].
- **Modulárne a integračné testovanie –** Je vykonávané pomocou nástroja Cantata++ na cieľovej platforme. Cantata++ umožňuje otestovanie príkazového a podmienkového pokrytia kódu.
- **Systémové testovanie –** Je vykonávané pomocou TPZRM.

4.2.8 Prechodové kritéria

Táto kapitola rozpracováva bod „e“, t.j. Prechodové kritéria, z Popis plánu verifikácie softwaru.
Popisuje kritéria pre vstup do verifikačného procesu SW. Je potrebné verifikovať SW plány a štandardy, vysokoúrovňové požiadavky, nízkoúrovňové požiadavky, SW architektúru, zdrojový kód, spustiteľný kód a samotnú verifikáciu. Pri každej z týchto verifikácií je potrebné určiť vstupné kritéria, činnosti verifikácie a výstupné kritéria.

Verifikácia SW plánov a štandardov

Vstupné kr.: Sú vytvorené SW plány PACS, PZK, PKM, PVS, VPS, štandardy ŠPP, ŠPN, ŠPK a technická špecifikácia.

Aktivity: Sú verifikované nasledujúce časti: základný formát, korektnosť a úplnosť, zhoda s DO-178B, kompatibilita podľa 4.2.6 Verifikácia softwarových plánov a štandardov.

Výstupné kr.: SW plány PACS, PZK, PKM, PVS, VPS a štandardy ŠPP, ŠPN, ŠPK sú verifikované, nezrovnalosti sú zaznamenané a opravené.

Verifikácia vysokoúrovňových požiadaviek

Vstupné kr.: SW plány PACS, PZK, PKM, PVS, VPS a štandardy ŠPP, ŠPN, ŠPK sú verifikované.

Aktivity: Sú verifikované nasledujúce časti: základný formát, presnosť a konzistencia, súlad so systémovými požiadavkami, zlučiteľnosť s cieľovým počítačom, overiteľnosť, zhoda s ŠPP, trasovateľnosť k systémovým požiadavkám, presnosť algoritmov.

Výstupné kr.: VÚP sú verifikované, nezrovnalosti zaznamenané sú a opravené.

Verifikácia nízkoúrovňových požiadaviek

Vstupné kr.: VÚP sú verifikované.

Aktivity: Sú verifikované nasledujúce časti: základný formát, presnosť a konzistencia, súlad s VÚP, zlučiteľnosť s cieľovým počítačom, overiteľnosť, zhoda s ŠPP, trasovateľnosť k VÚP, presnosť algoritmov.

Výstupné kr.: NÚP sú verifikované, nezrovnalosti zaznamenané sú a opravené.

Verifikácia architektúry SW

Vstupné kr.: VÚP sú verifikované.

Aktivity: Sú verifikované nasledujúce časti: základný formát, konzistentnosť, súlad s VÚP, zlučiteľnosť s cieľovým počítačom, overiteľnosť, zhoda s ŠPP, trasovateľnosť k VÚP.

Výstupné kr.: Architektúra SW je verifikované, nezrovnalosti sú zaznamenané a opravené.

Verifikácia zdrojového kódu

Vstupné kr.: Zdrojový kód je implementovaný, architektúra SW a NÚP sú verifikované.

Aktivity: Sú verifikované nasledujúce časti: súlad s NÚP, súlad s architektúrou SW, overiteľnosť, zhoda s ŠPK, presnosť a konzistentnosť.

Výstupné kr.: Zdrojový kód je verifikované, nezrovnalosti sú zaznamenané a opravené.

Testovanie spustiteľného kódu

Vstupné kr.: Zdrojový kód a testovacie postupy sú verifikovaný, je vytvorený spustiteľný kód.

Aktivity: Sú verifikované nasledujúce časti: súlad s VÚP, robustnosť s VÚP, súlad s NÚP, robustnosť s NÚP, súlad s architektúrou SW, kompatibilita s cieľovým počítačom.

Výstupné kr.: Zdrojový kód je otestovaný, nezrovnalosti sú zaznamenané a opravené.

Verifikácia výsledkov procesu verifikácie

Vstupné kr.: Sú pripravené výsledky testovania zdrojového kódu, recenzie VÚP, NÚP, architektúry SW, SW plánov a štandardov.

Aktivity: Sú verifikované nasledujúce časti: správnosť testovacích procedúr, správnosť testovacích výsledkov a vysvetlení rozporov, dosiahnutie pokrytia VÚP testami, dosiahnutie pokrytia NÚP testami, dosiahnutie príkazového a podmienkového pokrytia kódu, dosiahnutie pokrytia toku riadenia.

Výstupné kr.: Verifikačný proces je úspešne ukončený.

4.2.9 Rozkladanie

Nebude vykonávané. Rozkladanie je nutnou podmienkou len úrovne kritickosti A. Rozkladanie je technika, pri ktorej sa dosahuje dôsledná izolácia funkčne nezávislých SW zložiek, aby sa tak izolovali chyby a zabránilo sa tak ich šíreniu. Touto technikou sa potenciálne znižuje úroveň úsilia vynaloženého pri procese verifikácie SW.

4.2.10 Predpoklady kompilátora

Táto kapitola rozpracováva bod „g“, t.j. Predpoklady kompilátora, z Popis plánu verifikácie softwaru. Pre kompiláciu zdrojového kódu je použitý prekladač COSMIC. Pri kompilácii sa neuplatňuje žiadna optimalizácia. Žiadne iné knižnice tretej strany nie sú použité. Vysledovanie strojového kódu do zdrojového kódu je náročná vzhľadom k zložitosti kompilátora, preto overenie správnosti vytvoreného kódu sa bude vykonávať na základe testov.

4.2.11 Pokyny pre opätovné overovanie

Táto kapitola rozpracováva bod „h“, t.j. Pokyny pre opätovné overovanie, z Popis plánu verifikácie softwaru. Je potrebné stanoviť rozsah analýz a skúšok, ktoré musia byť zopakované, ak bol zmenený SW, ktorý už bol predtým testovaný. K tomu sa používajú aj napr. tieto metódy:

- Analýza dátového toku: skúma vplyv zmien na tok dát medzi jednotlivými funkciami a komponentmi SW.
- Analýza toku riadenia: skúma vplyv zmien v oblasti kontroly toku inštrukcií SW.
- Analýza časovania, nadväznosti, IO a ďalšie.

4.2.12 Predošle vyvinutý software

Neplatí pre náš projekt, pretože vyvíjaný SW neobsahuje žiadne časti predošle vyvinutého SW, teda SW je vyvíjaný tzv. „od nuly“.

4.2.13 Software s viacerými verziami

Neplatí pre náš projekt, pretože náš SW je určený len pre zariadenie jedného typu a aktuálna je vždy len jedna verzia.

5 Použitie postupu verifikácie SW v praxi

Táto kapitola sa venuje praktickej realizácii navrhnutého postupu verifikácie z kapitoly 4. Cieľom praktickej realizácie navrhnutého verifikačného postupu nie je kompletná realizácia všetkých aspektov verifikačného procesu, pretože nie je možné v rámci rozsahu diplomovej práce takúto činnosť realizovať. Cieľom realizácie verifikačného postupu je poskytnúť názornú ukážku toho, ako by mali byť požiadavky nadefinované, hierarchicky rozložené, ako by mali vyzerať rôzne druhy testovacích postupov a ich následná implementácia. Cieľom tejto kapitoly je poskytnúť také príklady, ktoré by z čo najväčšej časti pokryli rôznorodosť činností verifikačného procesu.

5.1 Požiadavky

Táto kapitola rozoberá spôsob ukladania, zobrazovania a definovania požiadaviek, ďalej tvorbu matíc trasovateľnosti a diagramov stromového zobrazenia požiadaviek. Proces tvorby požiadaviek a ich vlastností je popísaný v [3, 4]. Pri praktickej realizácii je vybraná iba vzorka systémových požiadaviek, z ktorých boli nedefinované požiadavky nižších úrovní a následne testovacie postupy, ktoré boli zrealizované. Konkrétne bola vybraná funkcia Štart, a teda všetky činnosti, ktoré súvisia s požiadavkou na štart motora.

Požiadavky boli ukladané v XML formáte. Pre užívateľsky príjemné zobrazenie sú XML súbory prevádzané do HTML formátu. Postup prevodu je následný:

0. Pomocou DTD súborov je overená syntax XML súborov.
1. Pomocou XSLT procesoru SAXON sú XML a XSL súbory prevedené do SDB formátu „docbooku“.
2. Pomocou aplikácie sdb2html sú prevedené SDB súbory do HTML formátu.

Identifikátory

Je potrebné, aby každá požiadavka mala svoj jedinečný identifikátor, spôsob jeho kódovania bol nasledovný: ZRM_„typ”_ „oblasť“_ „ID“. Pričom „typ“ mohol byť hodnoty TS pre systémové požiadavky, VUP pre vysokoúrovňové požiadavky a NUP pre nízkoúrovňové požiadavky. Premenná „oblasť“ určuje, z akej oblasti je daná požiadavka odvodená, napríklad pri požiadavkách na činnosti pri štarte motora, má táto premenná hodnotu START. Premenná „ID“ má číselnú podobu pre jedinečnosť identifikátoru.

Taktiež je potrebné definovať identifikátory testovacích postupov a ukazovateľov na miesta vo zdrojovom kóde. Tieto identifikátory majú rovnaký formát ako požiadavky, s tým, že premenná „typ“

má hodnotu ST pre systémový test, IT pre integračný test, MT pre modulárny test a KOD pre ukazovateľ na zdrojový kód.

Implementácia

Požiadavky sú definované vo formáte:

```
<req:req id="ZRM_VUP_START_1">
   <req:title>Odblokovanie a zablokovanie ventilu paliva</req:title>
   <req:owner>&lm;</req:owner>
   <req:revision>0</req:revision>
   <req:def>
      <para>
           Ventil paliva je možné softwarovo odblokovať a
           zablokovať.
      </para>
   </req:def>
   <req:comment></req:comment>
</req:req>
```

Pričom platí:

- Tag `title` definuje názov požiadavky.
- Tag `owner` definuje vlastníka požiadavky, v danom príklade má premenná lm hodnotu Lukáš Mačišák.
- Tag `revision` definuje číslo revízie požiadavky.
- Tag `def` definuje znenie požiadavky.
- Tag `comment` definuje prípadný komentár k požiadavke.

Ďalej je potrebné definovať status požiadavky, táto definícia sa uvádza do textového súboru status.txt, kvôli prehľadnosti, v tomto súbore je tiež možné okrem statusu uviesť rovnako aj komentár k danej požiadavke. Príklad obsahu súbor status.txt:

```
# ##############################################
# POPIS:
#  Tento súbor obsahuje stav jednotlivých systémových požiadaviek a
#  prípadne komentár k požiadavke. Stav je definovaný pomocou
#  statusu, ktorý môže nadobúdať nasledujúce hodnoty:
#     P - Navrhnutý (proposed)
#     A - Schválený (approved)
#     I - Implementovaný (implemented)
#     V - Overený (verified)
#     F - Odložený (deferred)
```

```
#      D - Zmazaný (deleted)
#      R - Zamietnutý (rejected)
# ##############################################
#   ID Požiadavky        Status          Komentár
# ------------------------------------------------------------
    ZRM_VUP_START_1       I
    ZRM_VUP_START_2       V
```

Taktiež je potrebné nadefinovať cesty k implementácii (má formu ďalších požiadaviek alebo zdrojového kódu) a cesty k testovacím postupom danej požiadavky. Táto definícia má nasledovný formát:

```
<req:traceinfo id="ZRM_TS_START_1">
      <req:implementation>
            <req:srcref refid="ZRM_VUP_START_13"/>
            <req:srcref refid="ZRM_VUP_START_15"/>
        </req:implementation>
        <req:testcase>
            <req:srcref refid="ZRM_ST_START_1"/>
        </req:testcase>
</req:traceinfo>
```

Pričom platí:

- Tag `traceinfo` definuje požiadavku, ktorej referencie sú definované.
- Tag `implementation` definuje odkazy, kde je požiadavka implementovaná.
- Tag `testcase` definuje odkazy na testovacie postupy danej požiadavky.

Výsledné HTML dokumenty požiadaviek sú vytvorené tak, aby užívateľ mal možnosť „preklikať“ sa od systémových požiadaviek cez vysokoúrovňové a nízkoúrovňové až po referenciu na zdrojový kód alebo testovací postup.

Všetky vytvorené požiadavky a vygenerované HTML súbory sú uvedené v prílohe 4.

Systémové požiadavky

Systémové požiadavky sú požiadavky najvyššej úrovne. Tieto požiadavky sú vytvorené z technickej špecifikácie, ktorá bola definovaná zákazníkom. Tieto požiadavky sú realizované vysokoúrovňovými požiadavkami a testované systémovými testami. Príklad výslednej vytvorenej požiadavky:

[ZRM_TS_START_1] Činnosti po príkaze k štartu motora

Po príkaze k štartu motora sa musia vykonať nasledujúce činnosti:

1. Odblokuje sa ventil paliva a zapaľovania.
2. Odstredivka oleja je uvedená do chodu. Otáčky odstredivky oleja sú nastavené na úroveň 20 000 ot/min.
3. Štartgenerátor je uvedený do chodu. Štartegenerátor postupne akceleruje až na otáčky, ktoré odpovedajú maximálnemu výkonu štartgenerátoru.
4. Čaká sa, až otáčky štartgenerátoru presiahňu úroveň 4 000 ot/min. Tieto otáčky musia byť dosiahnuté do doby 5s.

Komentár:	*Žiadny*
Vlastník:	*Zákazník*
Status:	*rev. 0 - Overený*
Implementácia:	*ZRM_VUP_START_13*
Test. postup:	*ZRM_ST_START_1*

Vysokoúrovňové požiadavky

Tieto požiadavky sú vytvorené zo systémových alebo vysokoúrovňových požiadaviek. Tieto požiadavky sú realizované vysokoúrovňovými alebo nízkoúrovňovými požiadavkami a testované integračnými testami. Príklad výslednej vytvorenej požiadavky:

[ZRM_VUP_START_2] Odblokovanie a zablokovanie ventilu zapaľovania

Ventil zapaľovania je možné softwarovo odblokovať a zablokovať.

Komentár:	*Žiadny*
Vlastník:	*Lukáš Mačišák*
Status:	*rev. 0 - Implementovaný*
Implementácia:	*ZRM_NUP_START_1* *ZRM_NUP_START_2*
Test. postup:	
Uspokojuje:	*ZRM_VUP_START_13* *ZRM_VUP_START_14*

Nízkoúrovňové požiadavky

Tieto požiadavky sú vytvorené z vysokoúrovňových alebo nízkoúrovňových požiadaviek. Tieto požiadavky sú realizované nízkoúrovňovými požiadavkami alebo implementované v zdrojovom kóde a testované modulárnymi testami. Príklad výslednej vytvorenej požiadavky:

[ZRM_NUP_START_8] Činnosti po ustálení horenia

Musí byť implementovaná funkcia, ktorá po ustálení horenia vykoná nasledujúce činnosti:

1. Deaktivuje úlohu, ktorá vykonáva činnosti po ustálení horenia.
2. Aktivuje regulátor otáčok motora.
3. Aktivuje úlohu, ktorá čaká na dosiahnutie minimálnych otáčok motora.

Komentár:	*Žiadny*
Vlastník:	*Ferko Mrkvička*
Status:	*rev. 0 - Overený*
Implementácia:	*ZRM_KOD_START_4*
Test. postup:	*ZRM_MT_START_4*
Uspokojuje:	*ZRM_VUP_START_16*

Matice trasovateľnosti a diagramy požiadaviek

Taktiež sú generované aj matice trasovateľnosti. Tie majú formát tabuľky a sú dvojakého typu:

- Matice trasoveteľnosti od systémových požiadaviek, cez vysokoúrovňové a nízkoúrovňové požiadavky, až po zdrojový kód. Pri tomto zobrazení môže kontrolór skontrolovať, či každá požiadavka bola realizovaná.
- Matice trasovateľnosti k testovacím postupom. Pri tomto zobrazení môže kontrolór skontrolovať, či každá požiadavka má pridelený testovací postup.

Ukážka matice trasovateľnosti je na Obr. 6.

Taktiež sú generované aj diagramy stromov požiadaviek, kde užívateľ má možnosť prehľadne vidieť, ako sa dospelo od systémovej požiadavky až po jej implementáciu v zdrojovom kóde. Ukážka stromu požiadaviek je na Obr. 5.

```
ZRM_TS_START_4
+-- ZRM_VUP_START_16
+-- ZRM_VUP_START_10
+-- ZRM_VUP_START_11
+-- ZRM_VUP_START_12
|      +-- ZRM_NUP_START_9
|                +-- ZRM_KOD_START_5
+-- ZRM_NUP_START_8
|      +-- ZRM_KOD_START_4
+-- ZRM_NUP_START_9
+-- ZRM_KOD_START_5
```

Obr. 5: Strom požiadaviek.

Požiadavka	Súbor/Doc.	Umiestnenie	Implementácia
ZRM TS START 1	ZRM TS rev.0	2.1: Funkcia ŠTART	ZRM VUP START 13
ZRM TS START 2	ZRM TS rev.0	2.1: Funkcia ŠTART	ZRM VUP START 14
ZRM TS START 3	ZRM TS rev.0	2.1: Funkcia ŠTART	ZRM VUP START 15
ZRM TS START 4	ZRM TS rev.0	2.1: Funkcia ŠTART	ZRM VUP START 16

Obr. 6: Matica trasovateľnosti.

5.2 Testy a testovacie postupy

Táto kapitola sa venuje testovacím postupom pre rôzne typy požiadaviek a testom vykonaným podľa týchto postupov.

Testovacie postupy

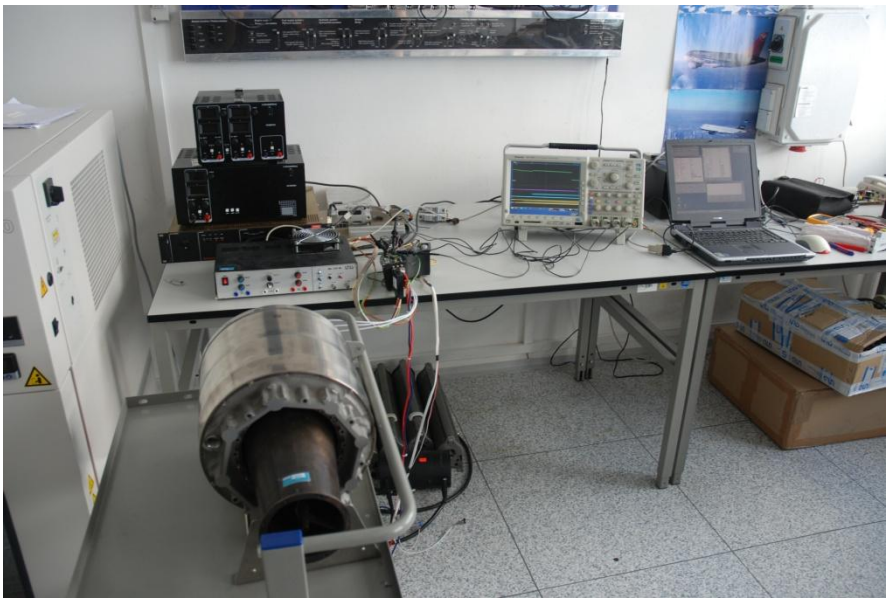
Každý vytvorený testovací postup obsahuje svoj jedinečný identifikátor, identifikátory požiadaviek ku ktorým sa vzťahuje, popis testu, prvotné inicializácie, postup testu a očakávané výsledky. Prakticky bol vytvorený a zrealizovaný jeden testovací postup pre systémové testovanie, jeden testovací postup pre integračne testovanie a šesť testovacích postupov pre modulárne testovanie. Tieto postupy sú uvedené v prílohe 4. Testovacie postupy vznikli na základe teoretických informácií uvedených v kapitole 4.2.5 Skúšobné metódy.

5.2.1 Systémové testovanie

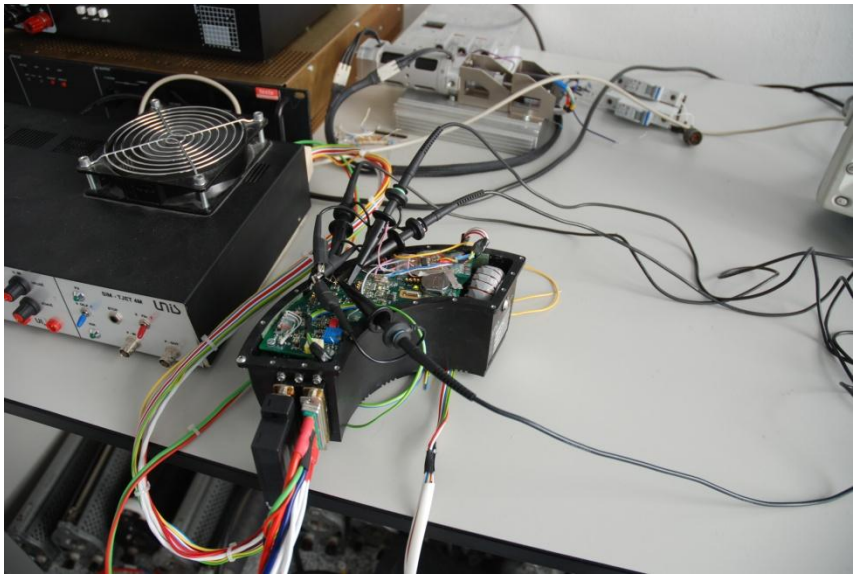
Systémové testovanie slúži na overenie splnenia systémových požiadaviek. V praxi bol realizovaný jeden systémový test typu testu funkčnosti, ktorý overoval dve systémové požiadavky. Testovací postup je uvedený v prílohe 4. Systémový test sa vykonáva na danom systéme, t.j. SW+HW. Pri tomto teste boli použité tieto zariadenia:

- TPZRM, ktorý simuluje odstredivku oleja a palivové čerpadlo.
- Prúdový motor TJ100 so štartgenerátorom.
- Osciloskop, na zistenie otáčok štartgenerátora, palivového čerpadla a odstredivky oleja. Taktiež je použitý na zistenie odblokovania ventilu paliva a zapaľovania, zopnutia ventilu paliva, čerpadla paliva a zapaľovania.
- Notebook s programom Monitor, ktorý komunikuje so ZRM po zbernici CAN.

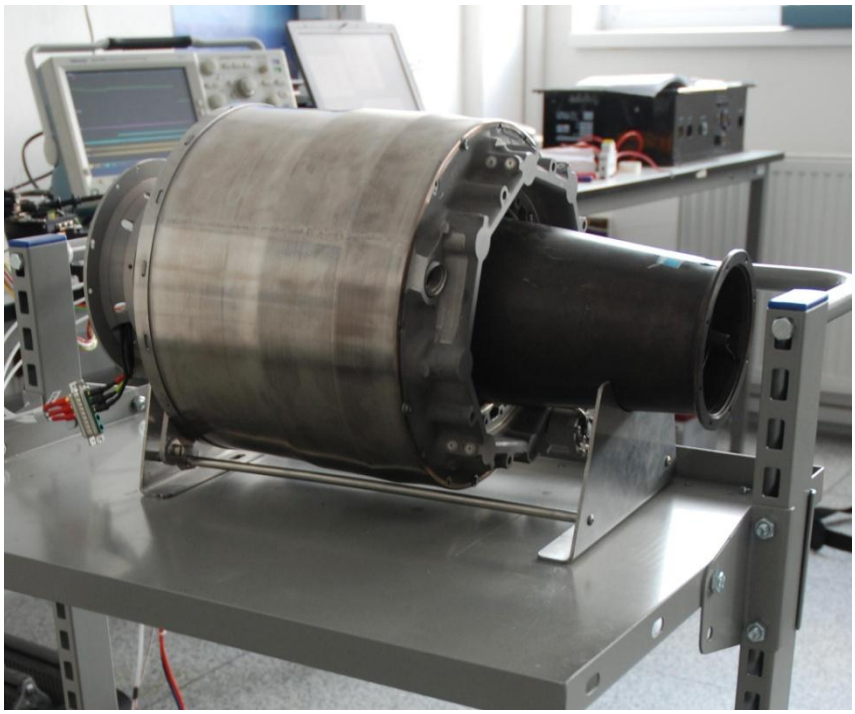
Dané zariadenia je možné vidieť aj na Obr. 7, 8 a 9.



Obr. 7: Systémový test – použité zariadenia.

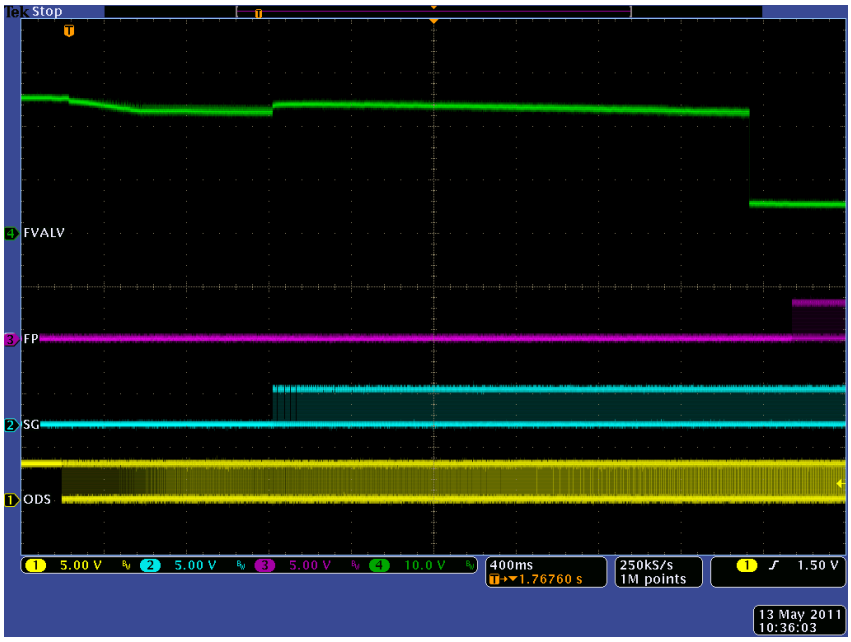


Obr. 8: Systémový test – ZRM.



Obr. 9: Systémový test – prúdový motor.

Pri tomto teste boli testované činnosti vykonané po príkaze k štartu motora, a to napr. nastavenie správnych otáčok štartgenerátora, odstredivky oleja a palivového čerpadla, zopnutie ventilov paliva, zapaľovania a palivového čerpadla v správnych časoch. Tento test prebehol úspešne, na Obr. 10 je možné vidieť časové priebehy otáčok štartgenerátora (priebeh SG), odstredivky oleja (priebeh ODS) a palivového čerpadla (priebeh FP). Bližší popis zhodnotenia nameraných výsledkov je možné nájsť v prílohe 4.



Obr. 10: Systémový test – namerané hodnoty.

5.2.2 Integračné testovanie

Integračné testovanie sa používa pri testovaní vysokoúrovňových a nízkoúrovňových požiadaviek. Pri tomto testovaní je potrebné najprv vytvoriť strom volania funkcií, ktorý bol vygenerovaný aplikáciou C Call-graph a časť z tohto stromu je uvedená v prílohe 3. Pre lepšiu názornosť je tento strom zobrazený aj pomocou diagramu uvedenom v prílohe 3. Na uvedený podstrom bol vytvorený jeden testovací postup uvedený v prílohe 4. Následne bol vytvorený testovací algoritmus v programe Cantata++.

Pri tomto testovaní je potrebné vytvoriť aj pomocný modul, v našom prípade pomenovaný ako helper, ktorý vytvára okolité prostredie testovaného kódu, slúži pre vytváranie vstupných dát a funkcií, s ktorými testovaný kód operuje.

Pri integračnom teste je jednou z hlavných úloh kontrola rozhrania medzi funkciou a blokmi kódu, ktoré sú do tejto funkcie integrované. Cantata++ umožňuje tieto rozhrania testovať na vysokej úrovni, umožňuje testovať hodnoty parametrov funkcií pri ich volaní a po návrate z volanej funkcie, taktiež umožňuje testovať návratovú hodnotu. V našom prípade sa testujú rozhrania medzi funkciou start_prep() a volanými funkciami. Rozhrania nižších úrovní sa už netestujú, predpokladá sa, že ich funkčnosť už bola otestovaná.

Ďalej je dôležité kontrolovať vykonanie všetkých požadovaných koncových udalostí, to znamená, že je potrebné otestovať výstupnú činnosť všetkých spolupracujúcich modulov (v našom prípade sú to moduly err, devcom, gpio, atď.). K daným modulom sa pristupuje ako k čiernym skrinkám. Cantata++ umožňuje porovnávanie rôznych typov premenných, volanie testovaných funkcií a podobne, pričom umožňuje aj volanie statických funkcií a porovnávanie statických premenných.

Pri tomto teste bola odhalená chyba rozhrania jednej z funkcií, bližší popis je uvedený vo výsledku testu uvedeného v prílohe 4.

5.2.3 Modulárne testovanie

Modulárne testovanie sa používa na overenie splnenia nízkoúrovňových požiadaviek. Pri modulárnom testovaní sa k funkciám pristupuje ako k bielym skrinkám, a teda je kontrolovaná jej vnútorná logika.

Pri tomto testovaní, ako aj pri integračnom testovaní, je potrebné vytvoriť aj pomocný modul, v našom prípade pomenovaný ako helper, ktorý vytvára okolité prostredie testovaného kódu, slúži pre vytváranie vstupných dát a funkcií, s ktorými testovaný kód operuje.

Testovacie algoritmy sú vytvárané pomocou programu Cantata++, ktorý umožňuje porovnávanie rôznych typov premenných, volanie testovaných funkcií a podobne, pričom umožňuje aj volanie statických funkcií a porovnávanie statických premenných.

Ďalej je dôležité zaručiť, že vnútorná logika daného modulu bola otestovaná v súlade s úrovňou kritickosti „B“ a teda, že pri testovaní bol celý modul podmienkovo a príkazovo pokrytý na 100%. Rovnako tento údaj o pokrytí je generovaný ako výstup programu Cantata++.

Pre modulárne testovanie bolo vytvorených päť testovacích postupov, ktoré boli implementované a vykonané. Zdrojové kódy a výsledky testov je možné nájsť v prílohe 4.

Analýza príkazového a podmienkového pokrytia kódu

Z vygenerovaného výstupu testu je možné vidieť, nie len percentuálne pokrytie, ale aj:

- Počty koľkokrát sa daný príkaz v zdrojovom kóde vykonal.
- Počty koľkokrát sa daná podmienka vyhodnotila pre stav pravda a koľkokrát pre stav nepravda.

Príklady výstupu sú uvedené na Obr. 11 a 12.

```
Summary by     EXECUTED     Overall                Statistics
Coverage type  INFEASIBLES  (wavg)     avg /    min /    max /    dev /    num
-------------------------------------------------------------------------------
statement              0    50.6%    33.3% /   0.0% / 100.0% /  48.7% /     15
decision               0    66.6%    93.3% /   0.0% / 100.0% /  25.8% /     15
-------------------------------------------------------------------------------
```

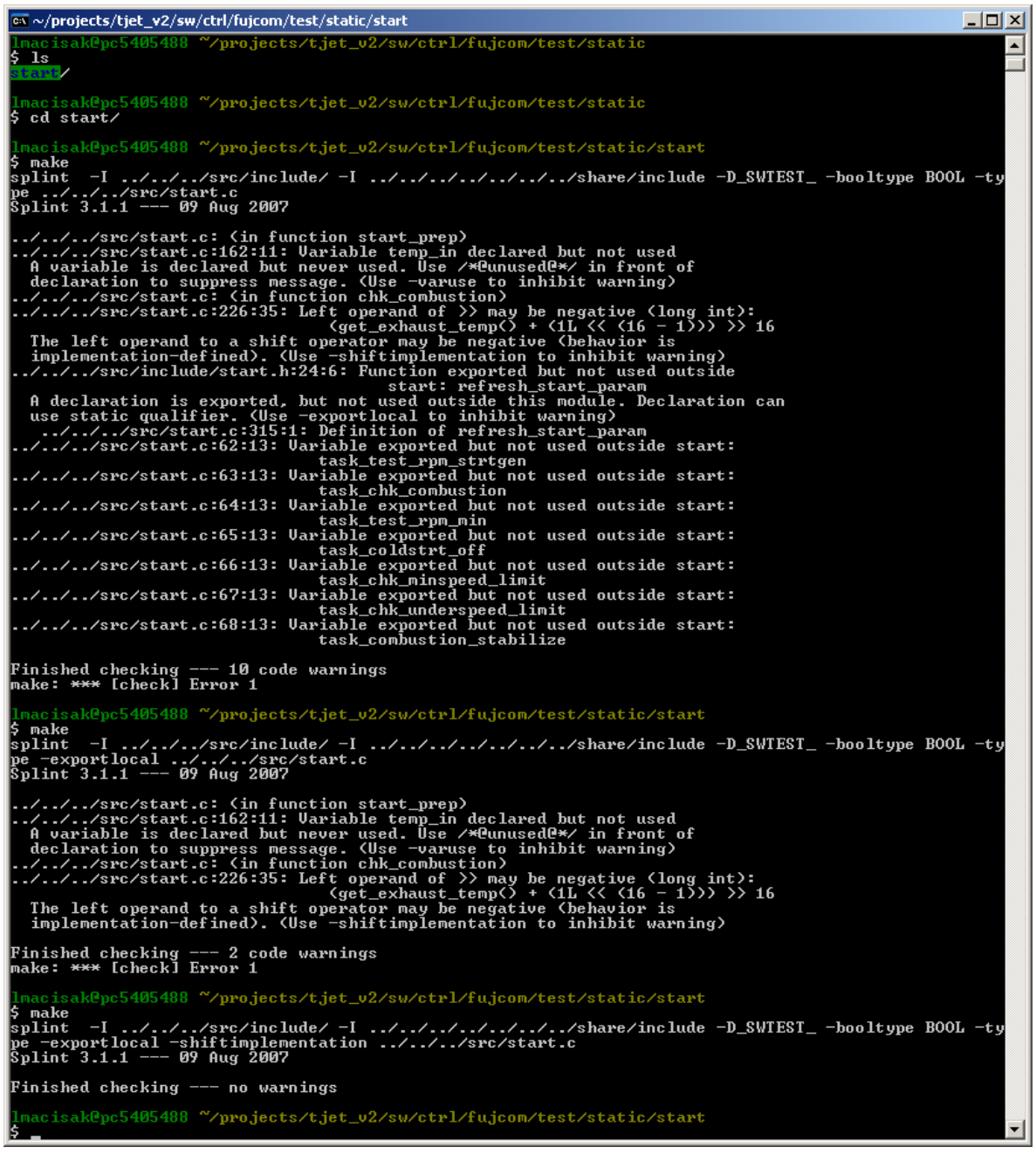
Obr. 11: Výstup analýzy príkazového a podmienkového pokrytia.

```
start.c(247):combustion_stabilize()
statement coverage details (with executed and un-executed cases)

start.c(249):               stmnt   1 (other)                              1
start.c(250):               stmnt   2 (other)                              1
start.c(251):               stmnt   3 (other)                              1

"combustion_stabilize"                                  executed           3
"combustion_stabilize"                                un-executed          0
"combustion_stabilize"                          statement coverage      100.0%

start.c(96):get_combustion_status()
statement coverage details (with executed and un-executed cases)

start.c(98):                stmnt   1 (return)              >> NOT EXECUTED

"get_combustion_status"                                 executed           0
"get_combustion_status"                               un-executed          1
"get_combustion_status"                         statement coverage        0.0%
```

Obr. 12: Analýza príkazového a podmienkového pokrytia.

5.2.4 Statická analýza

Statická analýza bola vykonaná pomocou voľne dostupného programu Splint. Makefile pre spustenie analýzy na module start a výsledok analýzy je uvedený v prílohe 4. Na Obr. 13 je možne vidieť výstup programu Splint. Z výstupu môžeme vidieť odhalenie nepoužitej premennej, ktorá bola odstránená. Ďalej upozornenie na rotáciu premennej, ktorá je znamienkového typu a taktiež nepoužitie globálne definovaných premenných, ktoré nie sú v iných moduloch použité, tieto hlásenia boli po prekontrolovaní potlačené.



Obr. 13: Statická analýza – Splint.

5.2.5 Výsledky testovania

Pri testovaní boli odhalené rôzne chyby, ako napríklad nepoužitá premenná, chyba v rozhraní, chybná implementácia jednej z funkcií. V prípade detekcie chyby je potrebné túto chybu popísať do výstupného formulára. Tento formulár je následne odovzdaný manažérovi projektu, ktorý na základe typov chýb oznámi nájdené chyby vývojárom SW. Vývojári SW danú chybu opravia alebo vyvrátia a upraví sa testovací postup. Následne sa daný test bude opätovne vykonávať,

5.3 Zhodnotenie postupu verifikácie a jeho praktickej realizácie

Táto kapitola sa venuje zhodnoteniu navrhnutého postupu a jeho realizácie, rozoberá jeho výhody a nevýhody, možnosti na zlepšenie.

V kapitole 4. Návrh postupu verifikácie SW pre konkrétnu aplikáciu je popísaný navrhnutý postup verifikácie SW. Tento postup bol vytvorený na základe požiadaviek normy DO-178B, na základe konzultácií s odborníkmi v odbore a na základe vlastných skúseností z praxe. Postup je napísaný takou formou, aby bolo možné jeho uplatnenie na akýkoľvek iný SW, ktorý má spĺňať požiadavky normy DO-178B pri jeho minimálnych úpravách. Avšak je vhodné použiť takýto postup aj pri realizácií bežného SW s hlavnou výhodou zvýšenia jeho kvality. Možnosti uplatnenia navrhnutého postupu v praxi sú široké. Postup je vytvorený tak, aby zaručil funkčnosť, bezpečnosť a kvalitu vyvíjaného SW. V prípade, že by mal byť tento postup aj schválený niektorou z certifikačných organizácií, je potrebné rozšíriť ho o referencie do ostatných základných dokumentov projektu, musí spĺňať formálne požiadavky a požiadavky konfiguračného manažmentu. Ďalej je potrebné verifikovať ho interne, v súlade s postupom uvedeným v kapitole 4.2.6 Verifikačné aktivity.

Pri realizácií verifikačného plánu je veľmi dôležitý výber použitých nástrojov a SW, z dôvodu časového a finančného hľadiska. Použité nástroje boli navrhnuté pre organizáciu, ktorá vyčlenila pätnástich ľudí na vývoj a verifikáciu daného zariadenia. Pre tvorbu požiadaviek boli navrhnuté nástroje a prostredie XML a HTML, ktoré sú zadarmo. Avšak pri projektoch väčšieho rozsahu je toto prostredie veľmi ťažko udržateľné v prehľadnej podobe, preto by bolo vhodné v budúcnosti zakúpiť alebo vyvinúť SW na základe databázy.

Pre tvorbu recenzií bolo navrhnuté prostredie textového editora MS Word. Ppäť pri projektoch väčšieho rozsahu je vhodné zakúpiť SW, ktorý by dané dokumenty prehľadne zobrazoval, ukladal a upozorňoval dané zodpovedné osoby na vzniknuté situácie.

Pre statickú analýzu bol navrhnutý nástroj Splint. Tento program je zdarma a jeho funkčnosť je dostatočná. Očakáva sa jeho používanie aj v budúcnosti.

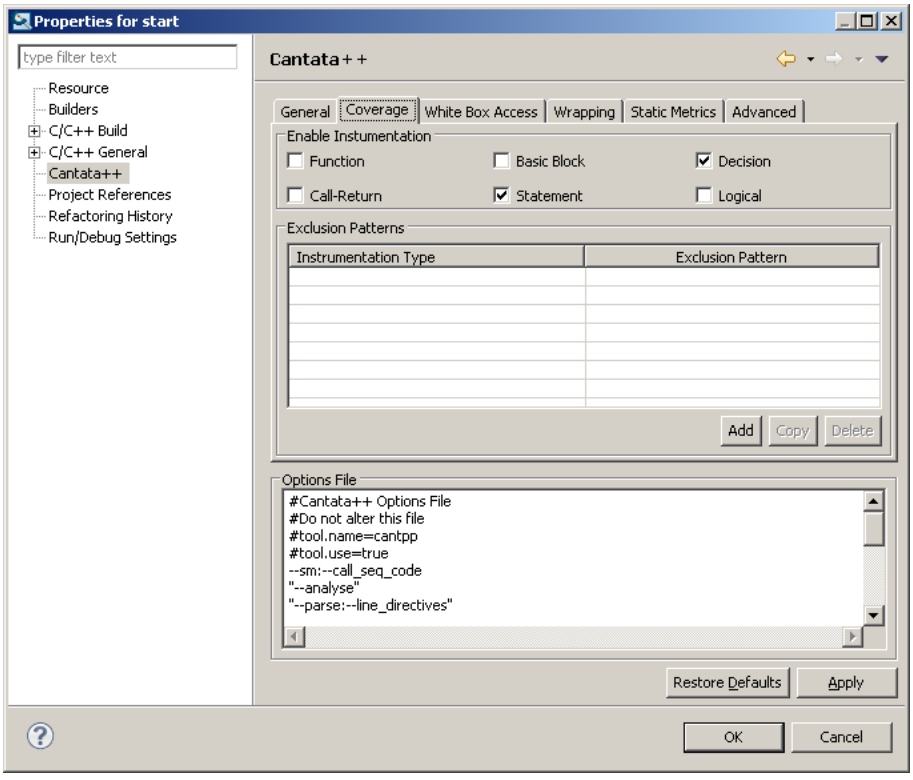
Cantata++

Pre integračné a modulárne testovanie bol navrhnutý nástroj Cantata++ (Obr. 15), cena tohto nástroja sa pohybuje do jedného milióna CZK. Nástroj Cantata++ je komplexný a navrhnutý priamo na testovanie SW. Umožňuje rôzne typy testov, ako sú testovanie formou biela a čierna skrinka, integračne, izolačné, komponentové testovanie a ďalšie. Cantata++ rovnako umožňuje podmienkové, príkazové a MCDC pokrytie kódu, pokrytie volania funkcií a ďalšie (Obr. 14). Umožňuje aj automatické generovanie testovacích skriptov, to sa však osvedčilo len pri jednoduchých zdrojových kódoch, v našom prípade táto možnosť nebola použitá. Ďalšou veľkou výhodou je možnosť spúšťania testov priamo na testovanom systéme, avšak aj táto možnosť si vyžaduje zakúpenie ďalších licencií a v našom prípade nebola zrealizovaná. Cantata++ umožňuje taktiež vývoj a ladenie SW.

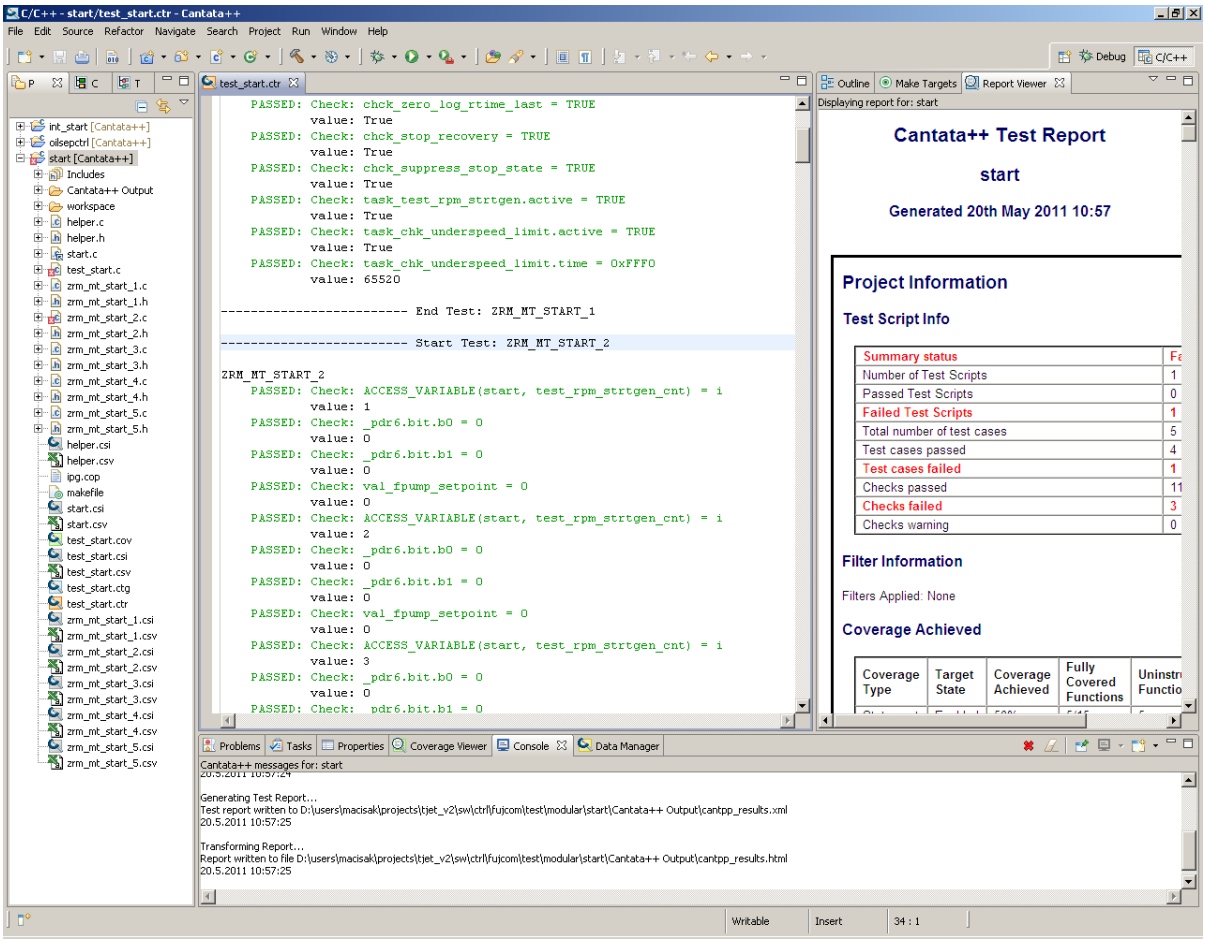
Jedným z cieľov tejto práce bolo aj tento nástroj naštudovať a prakticky použiť, čo sa podarilo a očakáva sa, že tento nástroj sa bude používať aj v budúcnosti, pri ďalších projektoch.

Avšak integračné a modulárne testovanie sa dá robiť aj bez akéhokoľvek špeciálneho nástroja, príkazová a podmienková analýza sa taktiež dá vykonávať aj manuálne. Napriek tomu sa oplatí investovať do týchto nástrojov, pretože len zistenie podmienkového a príkazového pokrytia manuálne by v projekte podobného rozsahu trvalo niekoľko človeko-rokov [3].

Je dobré si uvedomiť, že proces verifikácie je ako reťaz, ktorej sila je rovnaká ako sila najslabšieho článku, preto sa oplatí investovať do kvalitných verifikačných nástrojoch [3].



Obr. 14: Možnosti analýz pokrytia v Cantata++.



Obr. 15: Prostredie Cantata++.

6 Záver

Úlohou diplomovej práce bolo štúdium a analýza požiadaviek na certifikáciu a vývoj softwaru pre letectvo so zameraním na oblasť testovania a verifikácie v rámci životného cyklu projektu, návrh vlastného postupu verifikácie softwaru v zhode s požiadavkami normy RTCA/DO-178B a taktiež realizácia navrhnutého postupu na konkrétnej aplikácii.

Všetky body zadania boli v rozsahu diplomovej práce. Práca obsahuje štúdiu venujúcu sa testovaniu a verifikácii podľa danej normy. Detailne popisuje vybrané procesy životného cyklu softwaru, a to proces overovania softwaru a proces styku s certifikačným orgánom.

Práca ponúka taktiež návrh verifikačného plánu softwaru, podrobne vypracované jednotlivé body tohto plánu, a to konkrétne organizácia ľudí, pravidlá potrebnej nezávislosti pri verifikačných procesoch, metódy preskúmania, analytické a skúšobné metódy, verifikačné aktivity a overovacie prostredie. Okrem verifikačného plánu boli vytvorené aj kontrolné zoznamy, ktoré slúžia ako pomôcka pre uistenie, že boli vykonané všetky potrebné činnosti verifikácie.

Na základe verifikačného plánu bola v spolupráci s firmou UNIS a.s. vybraná konkrétna aplikácia. Z tejto aplikácie bola vybraná demonštračná časť, pre ktorú boli vytvorené systémové, vysokoúrovňové a nízkoúrovňové požiadavky. Pre niektoré z týchto požiadaviek boli vytvorené testovacie postupy a implementované modulárne a integračné testy. Na základe týchto postupov bolo následne vykonané systémové, modulárne a integračné testovanie. Cieľom realizácie verifikačného plánu bolo demonštrovať praktickú realizáciu čo najväčšieho množstva aspektov tohto plánu, tento cieľ považujem za splnený.

Verifikačný plán bol navrhnutý tak, aby bol znovu použiteľný aj na iných projektoch. V práci sú popísané možnosti na jeho zlepšenie a ďalšie použitie. Očakáva sa, že tento plán a jeho realizované časti budú v budúcnosti použité pri certifikácii vybranej aplikácie.

Problematika verifikácie a certifikácie kritického softwaru je rozsiahla a náročná na štúdium. Cieľom tejto práce bolo vytvoriť prehľad a názornú ukážku toho, čo všetko je pri týchto procesoch potrebné vykonávať. Aj tento cieľ považujem za splnený.

Literatura

[1] RTCA: *Software Considerations in Airborne Systems and Equipment Certification. RTCA/DO-178B*, December 1, 1992. Norma.

[2] RTCA: *Final Report for Clarification DO-178B „Software Considerations in Airborne Systems and Equipment Certification“. RTCA/DO-248B*, Október 12, 2001. Norma.

[3] Vance Hilderman; Tony Baghai: *Avionics certification. A Complete Guide to DO-178 (Software) DO-254 (Hardware)*. 2008, iSBN 978-1-885544-25-4.

[4] Christine Anderson; Merlin Dorfman: *Aerospace Software Engineering: Progress in Astronautics and Aeronautics.* Published by the American Institute of Aeronautics and Astronautics, 1991, iSBN 1-56347-005-5.

[5] Software Certification. [cit. 2010-15-12], [online].
URL <http://www.cigital.com/labs/reliability/certification.php>

[6] European Aviation Safety Agency. [cit. 2010-17-12], [online].
URL <http://en.wikipedia.org/wiki/European_Aviation_Safety_Agency>

[7] Life-critical system. [cit. 2010-17-12], [online].
URL <http://en.wikipedia.org/wiki/Life-critical_system>

[8] Safety Critical Systems. [cit. 2010-19-12], [online].
URL <http://www.comp.lancs.ac.uk/computing/resources/scs/>

[9] Capability Maturity Model (CMM). [cit. 2010-10-4], [online].
URL <http://searchsoftwarequality.techtarget.com/definition/Capability-Maturity-Model>

[10] ISO-9003:1997. [cit. 2010-10-4], [online].
URL <http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=26364>

[11] Department of Dedence, Washington: *DOD-STD-2167A: Military Standard, Defense System Software Development*. Februar 1988, [DVD-ROM]. Norma.

[12] MIL-STD-498. [cit. 2010-10-4], [online].
URL <http://www.everyspec.com/MIL-STD/MIL-TD+(0300++0499)/download.php?spec=MIL-STD-498.025500.pdf>

[13] ISO/IEC 12207:2008. [cit. 2010-10-4], [online].
URL <http://webstore.iec.ch/preview/info_isoiec12207%7Bed2.0%7Den.pdf>

[14] Splint. [cit. 2010-11-5], [online]. URL <http://www.splint.org/>

[15] Exida: *IEC 61508 Overview Report*. Januar 2006, [DVD-ROM]. Norma.

[16] Mauro Pezze; Michal Young: *Software Testing and Analysis: Process, Principles, and Techniques*. Daniel Sayre Publisher, 2008, iSBN-13 978-0-471-45593-6.

[17] Ilene Burnstein: *Pracitcal Software Testing: A Process-Oriented Approach. Springer-Verlag Publisher.* 2003, iSBN 0-387-95131-8.

[18] Aerospace short courses. [cit. 2010-15-4], [online]. URL <http://aeroshortcourses.ku.edu/>

Seznam příloh

Príloha 1. Doklady životného cyklu softwaru

Táto príloha obsahuje popis niektorých dokumentov, ktoré je potrebné vytvoriť počas životného cyklu softwaru. Väčšina spomenutých dokumentov je zameraná na proces overovania softwaru. Tieto a ďalšie doklady je možné nájsť v [1].

Popis plánu verifikácie softwaru

Plán overovania softwaru je popisom overovacích postupov slúžiacich k dosiahnutiu cieľov procesu overovania softwaru. Tieto postupy sa môžu odlišovať podľa úrovní kritickosti softwaru. Tento plán má obsahovať:

a) **Organizácia:** Organizačné zodpovednosti v procesu overovania softwaru a napojenie na ostatné procesy životného cyklu softwaru.
b) **Nezávislosť:** Popis metód pre zaistenie nezávislosti overenia, ak je táto nezávislosť požadovaná.
c) **Overovacie metódy:** Popis overovacích metód, ktoré budú použité pre každú jednotlivú činnosť procesu overovania softwaru:
    - Metódy preskúmania, vrátane kontrolných zoznamov alebo iných pomôcok.
    - Analytické metódy, vrátane rozboru trasovateľnosti a pokrytia.
    - Skúšobné metódy, vrátane pokynov pre výber skúšobných úloh, použitých skúšobných postupov a získaných údajoch.
d) **Overovacie prostredie:** Popis skúšobného zariadenia, skúšobných a analytických nástrojov a pokyny k použitiu týchto nástrojov a hardwaru skúšobného zariadenia.
e) **Prechodové kritéria:** Prechodové kritéria pre vstup do procesu overovania softwaru vymedzeného týmto plánom.
f) **Rozkladanie:** Metódy použité k overeniu neporušenosti (hraníc) rozkladu.
g) **Predpoklady kompilátoru:** Popis predpokladov, ktoré žiadateľ učinil ohľadom správnosti kompilátoru, spojovacieho programu alebo zavádzača.
h) **Pokyny pre znovu overovanie:** V prípade modifikácie softwaru, popis metód pre určenie (modifikáciu) zasiahnutých oblastí softwaru a pozmenených častí spustiteľného cieľového kódu. Znovu overovanie má zaručiť, že predošle hlásené chyby alebo triedy chýb boli odstránene.
i) **Predošle vyvinutý software:** V prípade použitia predošle vyvinutého softwaru a v prípade, kedy jeho bázová konfigurácia pôvodne podrobená overovaciemu procesu neodpovedá dokumentu DO-178b, potom popis metód vedúcich ku splneniu cieľov dokumentu DO-178b.

j) **Software s viacerými verziami:** V prípade použitia softwaru s viacerými verziami, potom popis činností procesu overovania softwaru.

Popis overovacích úloh a postupov

Tento doklad podrobne uvádza, ako sú realizované činnosti overovacieho procesu. Tieto údaje majú zahrňovať tieto informácie:

- **Postupy preskúmaní a analýz:** Podrobnosti, ktoré, ako doplnenie popisu uvedeného v Plánu overovania softwaru, popisujú rozsah a hĺbku metód preskúmania a analýz, ktoré budú použité.
- **Skúšobné úlohy:** Účel každej skúšobnej úlohy, možnosti vstupov, podmienky, očakávané výsledky vedúce k splneniu požadovaných kritérií pokrytia a kritéria typu „vyhovel/nevyhovel".
- **Skúšobné postupy:** Inštrukcie popisujúce krok za krokom, ako má byť každá jednotlivá skúšobná úloha zostavená a vykonaná, ako budú vyhodnotené jej výsledky a aké prostredie bude pre ňu použité.

Popis výsledkov overovania softwaru

Sú produktom činností procesu overovania softwaru. Výsledky overovania softwaru majú:

- U každého preskúmania, analýzy a skúšky označiť každý postup, ktorý v priebehu činností vyhovel resp. nevyhovel a uviesť konečný výsledok.
- Identifikovať konfiguračnú položku, ktorá bola podrobená preskúmaniu, analýze alebo skúške.
- Obsahovať výsledky skúšok, preskúmaní, analýz, vrátane analýz pokrytia a analýz trasovateľnosti.

Popis plánu softwarových aspektov osvedčovania spôsobilosti

Je to základný doklad, ktorý certifikačný orgán používa pri rozhodovaní o tom, či žiadateľ navrhuje taký životný cyklus SW, ktorý je úmerný vzhľadom k prísnosti požadovanej pre danú úroveň kritickosti vyvíjaného SW. Tento plán má zahŕňať:

- **Obecný popis systému:** Táto kapitola uvádza prehľad systému vrátane popisu jeho funkcií a ich rozdelenie medzi HW a SW, architektúry použitého procesoru, rozhrania HW/SW a bezpečnostných parametrov.
- **Obecný popis SW:** Táto kapitola stručne popisuje funkcie SW s dôrazom na navrhované koncepcie bezpečnosti.
- **Osvedčovanie spôsobilosti:** Táto kapitola poskytuje stručný prehľad predpisovej základne, vrátane tých spôsobov preukázania plnenia jej požiadaviek, ktoré sa dotýkajú softwarových

aspektov osvedčovania spôsobilosti. Táto kapitola taktiež uvádza stanovenú úroveň kritickosti SW a zhrňuje jej zdôvodnenie, ktoré priniesol proces posudzovania bezpečnosti SW.

- **Životný cyklus softwaru:** Táto kapitola definuje životný cyklus SW, ktorý má byť použitý a obsahuje stručný súhrn každého životného cyklu SW a všetkých jeho procesov, pre ktoré sú podrobné informácie uvedené v príslušných SW plánoch.
- **Doklady životného cyklu softwaru:** Táto kapitola špecifikuje doklady životného cyklu SW, ktoré budú vypracované a riadené procesmi životného cyklu SW.
- **Časový plán:** Táto kapitola popisuje spôsoby, ktoré žiadateľ použije, aby poskytol certifikačným orgánom možnosť nahliadnuť do činností procesu životného cyklu SW a umožnil mu plánovať jednotlivé preskúmania.
- **Doplnkové úvahy:** Táto kapitola popisuje špeciálne prostriedky, ktorých použitie môže ovplyvniť proces osvedčovania spôsobilosti, napr. alternatívne metódy dokazovania, kvalifikácia nástrojov, skôr vyvinutý SW, atď.

Príloha 2. Kontrolné zoznamy

Kontrolné zoznamy slúžia ako pomôcka pre recenzenta alebo inšpektora pri verifikácii. Cieľom kontrolných zoznamov je zabezpečiť, aby recenzent alebo inšpektor nezabudol na žiaden z aspektov verifikácie. Skratka A/N znamená áno/nie.

Kontrolný zoznam softwarových plánov a štandardov

SW plány a štandardy	
Popis	*A/N*
Základný formát	
Obsahuje dokument správne číslo verzie?	
Obsahuje titulná strana názov dokumentu, dátum vypracovania, identifikačný kód?	
Je prítomná a správna história revízií dokumentu?	
Sú všetky obrázky, tabuľky a diagramy správne označené?	
Sú plány popísané primeranou úrovňou detailnosti?	
Sú SW plány a štandardy korektné z formálneho hľadiska?	
Korektnosť a úplnosť	
Sú SW plány a štandardy kompletné?	
Sú SW plány a štandardy presné?	
Sú SW plány a štandardy uskutočniteľné?	
Sú SW plány a štandardy jednoznačné?	
Zhoda s DO-178B	
Sú SW plánmi a štandardami pokryté všetky ciele DO-178B?	
Sú SW plány a štandardy naviazané na DO-178B?	
Použité metódy sú v súlade s DO-178B?	
Kompatibilita	
Sú SW plány a štandardy konzistentné?	
Sú krížové odkazy medzi dokumentmi korektné?	
Majú SW plány jednotný časový harmonogram?	
Je popis životného cyklu v SW plánoch jednotný?	
Sú SW plány v súlade s technickou špecifikáciou?	
Sú použité metódy v SW plánoch jednotné?	
Je použité prostredie v SW plánoch jednotné?	
Sú definované zodpovednosti v SW plánoch jednotné?	

Kontrolný zoznam softwarových požiadaviek

SW požiadavky	
Popis	*A/N*
Základný formát	
Obsahuje dokument správne číslo verzie?	
Obsahuje titulná strana názov dokumentu, dátum vypracovania, identifikačný kód?	
Je prítomná a správna história revízií dokumentu?	
Sú všetky obrázky, tabuľky a diagramy správne označené?	
Má každá požiadavka jedinečný identifikátor?	
Presnosť a konzistentnosť	
Sú požiadavky definovane presne a zrozumiteľne?	
Sú požiadavky vzájomne bezkonfliktné?	
Sú definované odvodené požiadavky?	
Súlad s požiadavkami vyššej úrovne	
Sú požiadavky v súlade s dokumentmi, na základe ktorých vznikli?	
Sú požiadavky v súlade s výkonovými charakteristikami?	
Sú požiadavky v súlade s technickou špecifikáciou?	
Sú požiadavky v súlade s bezpečnostnými požiadavkami?	
Sú požiadavky v súlade s požiadavkami kvality?	
Sú požiadavky v súlade s definíciou systémových zdrojov?	
Zlučiteľnosť s cieľovým počítačom	
Sú definované požiadavky na inicializáciu?	
Sú definované požiadavky na činnosti pri vzniku chýb?	
Sú definované požiadavky na vstavané testovanie a monitorovanie systému?	
Obsahuje dokument popis alebo odkaz na popis detekcie HW chýb?	
Obsahuje dokument popis alebo odkaz na popis komunikačného rozhrania?	
Obsahuje dokument popis alebo odkaz na popis pamäti?	
Obsahuje dokument popis alebo odkaz na popis periférnych zariadení?	
Obsahuje dokument popis alebo odkaz na popis analógovo-digitálneho prevodu?	
Overiteľnosť	
Je každá požiadavka realizovateľná?	
Je každá požiadavka testovateľná? Je možné nezávisle overenie, či je daná požiadavka splnená?	
Súlad s ŠPP	
Sú požiadavky v súlade s požiadavkami ŠPP?	
Sú požiadavky správne rozdelené do kategórií?	
Sú požiadavky dostatočne podrobné?	
Nie sú požiadavky príliš podrobné?	
Trasovateľnosť k požiadavkám vyššej úrovne	
Je u každej požiadavky uvedený odkaz na jej zdroj, na základe ktorého bola vytvorená?	
Bola vykonaná analýza na identifikáciu chýbajúcich požiadaviek?	

Kontrolný zoznam návrhu algoritmov

Návrh algoritmov	
Popis	*A/N*
Návrh algoritmov	
Je dosiahnutá dostatočná úroveň presnosti algoritmov?	
Je algoritmus navrhnutý pre celý rozsah vstupných dát?	
Sú vstupné a výstupné rozsahy dát vymedzené?	
Zodpovedajú dynamické vlastnosti algoritmu očakávaným?	

Kontrolný zoznam architektúry softwaru

Architektúra SW	
Popis	*A/N*
Základný formát	
Obsahuje dokument správne číslo verzie?	
Obsahuje titulná strana názov dokumentu, dátum vypracovania, identifikačný kód?	
Je prítomná a správna história revízií dokumentu?	
Sú všetky obrázky, tabuľky a diagramy správne označené?	
Sú všetky termíny a jednotky meraní vhodné?	
Konzistentnosť	
Je popis architektúry SW presný a zrozumiteľný?	
Nepoužívajú sa rôzne pojmy pre tú istú vec?	
Nepoužívajú sa rovnaké pojmy pre rôzne veci?	
Súlad s VÚP	
Sú špecifikované modely, algoritmy a numerické techniky kompatibilné s VÚP?	
Sú presnosti a jednotky vstupov a výstupov kompatibilné s VÚP?	
Sú VÚP na externé rozhrania kompatibilné s architektúrou SW?	
Sú VÚP na interné rozhrania kompatibilné s architektúrou SW?	
Zlučiteľnosť s cieľovým počítačom	
Je architektúra SW v súlade s architektúrou CPU?	
Je v súlade veľkosť, typ a adresný priestor pamätí s architektúrou SW?	
Sú vyrovnávacie pamäte a fronty riadené a ohraničené?	
Sú riadiace frekvencie a frekvencie príjmu dát v súlade so systémovými zdrojmi?	
Overiteľnosť	
Je architektúra SW realizovateľná?	
Je architektúra SW testovateľná?	
Je architektúra SW špecifikovaná natoľko jasne, aby bola zrozumiteľná pre nezávislého človeka/skupinu ľudí?	
Súlad s ŠPP	
Je architektúra SW v súlade s požiadavkami ŠPP?	
Je architektúra SW správne rozdelená do kategórií?	
Je architektúra SW dostatočne podrobná?	
Nie je architektúra SW príliš podrobná?	

Kontrolný zoznam algoritmov

Algoritmy	
Popis	*A/N*
Algoritmy	
Sú hodnoty algoritmu obmedzené?	
Sú statické premenné inicializovane správne?	
Je počiatočný stav algoritmu správny?	
Je analyzovaný vplyv chýb zaokrúhlením?	
Sú použité správne dátové typy pre ukladanie dát?	
Je v definovaných rozsahoch zaručená kontinuita dát?	

Kontrolný zoznam recenzie zdrojového kódu

Zdrojový kód - recenzia	
Popis	*A/N*
Súlad s NÚP	
Je zdrojový kód v súlade s dokumentmi, na základe ktorých vznikol?	
Je zdrojový v súlade s výkonovými charakteristikami?	
Je zdrojový v súlade s definíciami rozhraní?	
Je zdrojový v súlade s bezpečnostnými požiadavkami?	
Je zdrojový v súlade s definíciou systémových zdrojov?	
Súlad s architektúrou SW	
Je štruktúra zdrojového kódu v súlade so štruktúrou architektúry SW?	
Je zdrojový kód rozdelený správne do vrstiev?	
Sú implementované dátové toky v súlade s architektúrou SW?	
Sú implementované toky riadenia v súlade s architektúrou SW?	
Overiteľnosť	
Boli pri implementácií použité také postupy a techniky, aby bolo možné zdrojový kód jednoducho testovať?	
Je zdrojový kód implementovaný natoľko jasne, aby bola zrozumiteľná pre nezávislého človeka/skupinu ľudí?	
Súlad s ŠPK	
Je zdrojový kód v súlade s ŠPK?	
Trasovateľnosť k NÚP	
Je v celom zdrojovom kóde uvedený odkaz na NÚP, na základe ktorého bola daná časť implementovaná?	
Presnosť a konzistentnosť	
Je zdrojový kód kompletný?	
Sú rozhrania implementované správne?	
Sú implementované parametre a ich dátové typy správne?	

Kontrolný zoznam inšpekcie zdrojového kódu

Zdrojový kód - inšpekcia	
Popis	*A/N*
Chybné referencie dát	
Sú všetky premenné, ktoré sú referencované pomocou ukazateľov, inicializované?	
Sú indexy do polí a reťazcov celočíselného typu a sú vždy v medziach rozsahu daného poľa alebo reťazca?	
Je všade na "vhodných" miestach, napr. kontrola hranice poľa, použitá konštanta namiesto premennej?	
Je premenným vždy priradená hodnota rovnakého typu ako je typ premennej?	
Ukazuje každý ukazateľ na alokované pamäťové miesto?	
Chybné deklarácie dát	
Je každá premenná správneho dátového typu?	
Ak je premenná pri deklarácii aj inicializovaná, je inicializovaná správne a v súlade s jej dátovým typom?	
Odkazuje na každú deklarovanú premennú maximálne jeden ukazateľ?	
Chyby výpočtov	
Sú premenné pri určitom výpočte rovnakého dátového typu (napr. celočíselného)?	
Sú premenné pri určitom výpočte rovnakého veľkého dátového typu (napr. 1 byte)?	
Nie je možné pretečenie alebo podtečenie niektorej numerickej metódy?	
Nie je možné aby nastalo delenie nulou?	
Je strata presnosti pri niektorých matematických operáciách akceptovateľná?	
Chyby porovnaní	
Sú porovnania implementované správne? (napr. chyba typu menšie vs. menšie alebo rovné)	
Porovnávajú sa reálne čísla s akceptovateľnou presnosťou?	
Sú boolovske výrazy implementované správne? (napr. chybné umiestnenie zátvoriek)	
Používajú sa v boolovskych výrazoch len premenné typu Boolean?	
Chyby toku riadenia	
Sú všetky moduly, podprogramy alebo slučky ukončiteľné? Ak nie, je to akceptovateľne?	
Je možné, že sa niektorá slučka nevykoná ani raz? Ak áno, je to akceptovateľné?	
V pripade implementácie "switch - case", môže premenná na základe ktorej sa vyberá "case" nadobúdať aj hodnotu pre ktorú neexistuje "case"? Ak áno, je tento prípad patrične ošetrený?	
Chyby parametrov podprogramov	
V prípade volania podprogramov, je poradie parametrov správne?	
V prípade volania podprogramov, odpovedajú dátové typy parametrov deklarácií podprogramu?	
Ak sú ako parametre odovzdané konštanty, je ich hodnota v podprograme modifikovaná? Ak áno je to akceptovateľné?	
Vstupno/výstupné chyby	
Je formát dát, ktoré sa prijímajú alebo odosielajú na externé zariadenia správny?	

Ak externé zariadenie nie je prítomné alebo pripravené, je tento stav ošetrený?	
Sú ošetrené všetky možné chyby, ktoré môžu nastať na externom zariadení?	
Sú všetky chybové hlásenia správne z hľadiska významu, vhodnosti a gramatiky?	

Kontrolný zoznam kompatibility softwaru s cieľovým hardwarom

Kompatibilita SW s cieľovým HW	
Popis	*A/N*
Kompatibilita SW s cieľovým HW	
Je spustiteľný cieľový kód kompatibilný s cieľovým HW?	
Je umiestnenie pamäťových segmentov kompatibilné s cieľovým HW?	
Sú veľkosti pamäťových segmentov menšie alebo rovnako veľké ako fyzické pamäťové segmenty?	
Sú adresy registrov správne namapované?	

Kontrolný zoznam výsledkov testovania

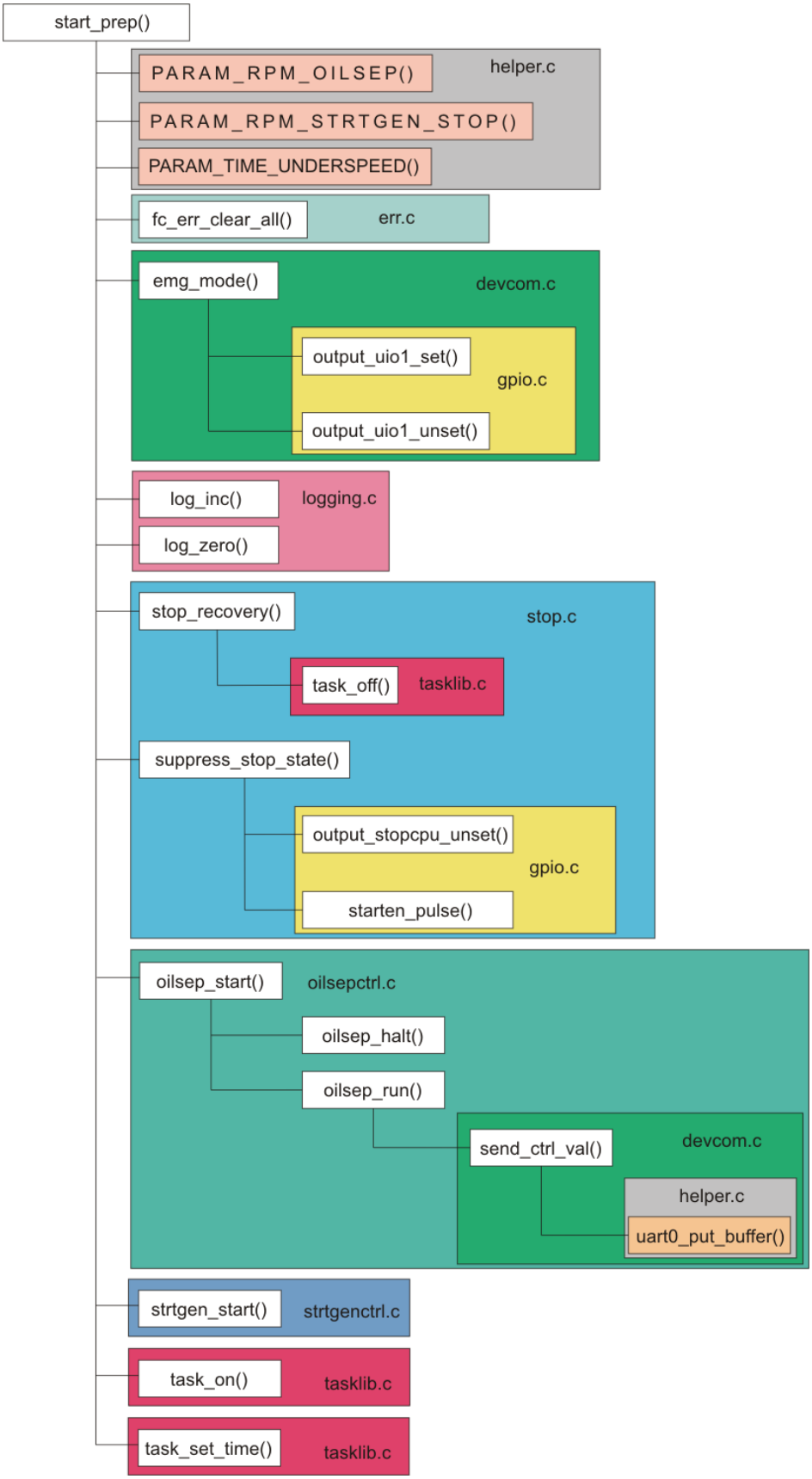
Výsledky testovania	
Popis	*A/N*
Správnosť testovacích procedúr	
Vzťahuje sa každá testovacia procedúra na niektorú z požiadaviek?	
Sú skúšobné postupy realizovateľné?	
Sú skúšobné postupy formálne správne?	
Správnosť testovacích výsledkov a vysvetlení rozporov	
Boli vykonané všetky testovacie procedúry?	
Sú všetky nezrovnalosti ohlásené a zdokumentované?	
Sú vykonané korekcie prípadne vypracované vysvetlenia všetkých nezrovnalostí?	
Je vykonaná opätovná verifikácia po vykonaní zmien?	

Príloha 3. Integračné testovanie

Ukážka stromu volania funkcií

start_prep() <void start_prep (void) at start.c:160>:
    PARAM_RPM_OILSEP_UINT32()
    PARAM_RPM_STRTGEN_STOP_UINT32()
    fc_err_clear_all() <void fc_err_clear_all (void) at err.c:105>:
    emg_mode() <void emg_mode (BOOL state) at devcom.c:53>:
        output_uio1_set()
        output_uio1_unset()
    log_inc() <void log_inc (UBYTE event) at logging.c:173>:
    log_zero() <void log_zero (UBYTE event,ULONG val) at logging.c:125>:
    stop_recovery() <void stop_recovery (void) at stop.c:367>:
        task_off()
    suppress_stop_state() <void suppress_stop_state (void) at stop.c:388>:
        output_stopcpu_unset()
        starten_pulse() <void starten_pulse (void) at gpio.c:150>:
    task_set_time()
    PARAM_TIME_UNDERSPEED_UINT16()
    oilsep_start() <LONG oilsep_start (LONG sp) at oilsepctrl.c:181>:
        oilsep_halt() <void oilsep_halt (void) at oilsepctrl.c:141>:
            SET_OILSEP_CTRL()
        is_task_on()
        oilsep_run() <void oilsep_run (void) at oilsepctrl.c:153>:
            SET_OILSEP_CTRL()
            PARAM_YSTRT_OSEP_UINT16()
    strtgen_start()<LONG strtgen_setpoint (LONG sp) at strtgenctrl.c:246>:
        strtgen_halt() <void strtgen_halt (void) at strtgenctrl.c:215>:
            SET_STRTGEN_CTRL()
        is_task_on()
        strtgen_run() <void strtgen_run (void) at strtgenctrl.c:203>:
            calc_ctrl_start() <UINT16 calc_ctrl_start (void) at strtgenctrl.c:186>:
                PARAM_YSTRT_STRTGEN_UINT16()
    task_on()
    task_off()

Diagram volania funkcií



Príloha 4. Obsah DVD

./technicka_sprava/ - technická správa (tento dokument) vo formáte .pdf

./technicka_sprava/src - technická správa (tento dokument) vo formáte .doc

./citovane_dokument/ - citované dokumenty

./prakticka_cast/poziadavky/nup/ - nízkoúrovňové požiadavky vo formáte .html a .xml

./prakticka_cast/poziadavky/vup/ - vysokoúrovňové požiadavky vo formáte .html a .xml

./prakticka_cast/poziadavky/pts/ - požiadavky technickej špecifikácie vo formáte .html a .xml

./prakticka_cast/poziadavky/matice_trasovatelnosti/ - matice trasovateľnosti vo formáte .html a .xml, vygenerované stromy požiadaviek vo formáte .html

./prakticka_cast/testy/integracne/ - vytvorené postupy pre integračné testovanie spolu s výsledkami testov, zdrojové kódy testov

./prakticka_cast/testy/modularne/ - vytvorené postupy pre modulárne testovanie spolu s výsledkami testov, zdrojové kódy testov

./prakticka_cast/testy/systemove/ - vytvorené postupy pre systémové testovanie spolu s výsledkami testov a obrázkami dokumentujúcimi priebeh testovania

./prakticka_cast/testy/zdrojovy_kod/ - zdrojový kód na ktorom bolo vykonávané testovanie