Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

RAČUNALNIŠKA ZANESLJIVOST IN DIAGNOSTIKA
SEMINARSKA NALOGA

ZANESLJIVOSTNA ANALIZA
SAMSUNG GALAXY TAB

Božić Darko,
Krajačić Ivan,
Živec Marko,
Cesar Anže,
Sakelšak Dejan,
Škaper Gregor,
Andrejak Luka,
Cvenkel Gašper,
Jež Jani,
Struna Simon,
Berdajs Jan,
Bizjak Matevž,
Krečič Marko,
Adzievski Pece,
Paspalovski Borče,
Uršič Aleš.

Ljubljana, maj 2011

Kazalo vsebine

1. Uvod (Krajačić)

Pri predmetu Računalniška zanesljivost in diagnostika smo za seminarsko nalogo naredili zanesljivostno analizo tabličnega računalnika Samsung Galaxy TAB. Našo zanesljivostno analizo sestavljajo:

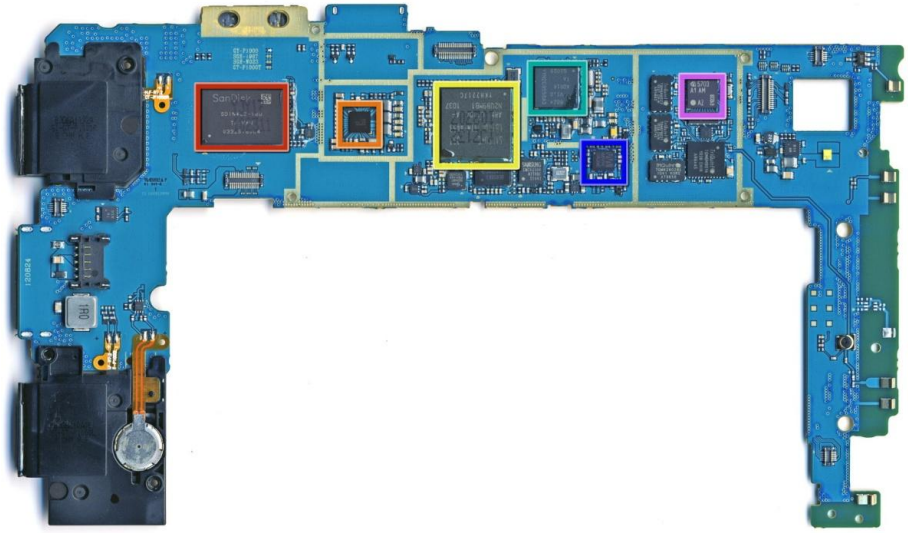
- Zanesljivost strojne opreme izračunana po MIL 217F in IEC 62380 standardu
- Analiza programske opreme s pomočjo metrik kompleksnosti programske opreme
- Analiza drevesa napak – FTA (angl. fault tree analysis)
- Analiza načinov in učinkov odpovedi – FMEA (angl. failure modes and effects analysis)
- Markovska analiza

1.1. Tehnične značilnosti Samsung Galaxy TAB

Slika 1.1: Samsung Galaxy TAB [1.1]

Procesor :	Samsung S5PC110A01 (1GHz ARM Cortex A8 Core, PowerVR SGX540 grafika)
Operacijski sistem :	Android OS 2.2 (Froyo) v slovenščini
Spomin :	Flash: 16GB, RAM: 512MB
Dimenzije :	190.1 x 120.5 x 12 mm
Teža :	380 gramov
Zaslon :	7.0" na dotik, 600 x 1024 točk, 16 milijonov barv
Omrežja :	HSUPA 5,76/HSDPA 7,2Mbps 900/1900/2100 MHz, GSM/GPRS/EDGE: 850/900/1800/1900 MHz, Wi-Fi a/b/g/n, BT3.0, DLNA
GPS :	A-GPS
Kamera :	3.0 MP z avtofokusom, LED flash + 1.3 MP
Baterija :	Li-Poly, 4000 mAh
Razširitvena reža :	microSD do32GB

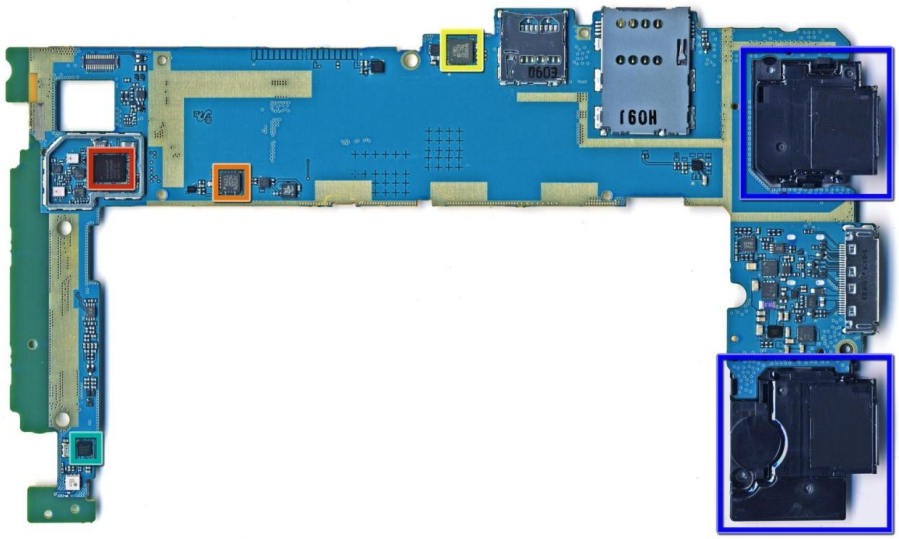
1.2. Osnovne komponente



Slika 1.2: PCB sprednja stran [1.2]

Na sliki 1.2 imamo označene osnovne komponente na sprednji strani tiskanega vezja, to so:

- Samsung S5PC110A01 + Samsung KB100D100YM
- Maxim 8998 (Power Management IC)
- SanDisk SDIN4C2-16G (MLC NAND 16 GB Flash pomnilnik)
- Infineon PMB5703 (RF oddajnik)
- Infineon PM9801 (X-GOLD 616 HSDPA/HSUPA/EDGE Modem)
- Wolfson Microelectronics WM8994 (Audio Codec)



Slika 1.3: PCB zadnja stran [1.2]

Na sliki 1.3 imamo označene osnovne komponente na zadnji strani tiskanega vezja, to so:

- ATMEL MXT224 (Krmilnik za ekran na dotik)
- STMicroelectronics L3G4200D (Digitalni 3-D MEMS Gyroscope)
- Broadcom BCM4329 (Bluetooth/FM/WLAN sprejemnik in oddajnik)
- Broadcom BCM4751 (GPS sprejemnik)
- Zvočniki

2. Zanesljivost strojne opreme (Božić, Krajačić, Živec)

Naša naloga je bila, določiti zanesljivost tabličnega računalnika Samsung Galaxy TAB po MIL 217F [2.1] in IEC 62380 [2.8] standardu. Tako za MIL kot IEC smo morali najprej poiskati sestavne komponente te naprave. Napravo je dokaj težko razstaviti, ne da bi se kaj polomilo, zato smo se odločili, da je ne razstavimo, ampak poiščemo kar se da natančne podatke o samih sestavnih komponentah na internetu. Poiskali smo dokaj podrobne slike [1.2] že razstavljene naprave, kjer se razločno vidi katere so osnovne komponente. Prav tako smo poiskali tudi spisek vseh sestavnih komponent [2.2] in se odločili le te uporabiti pri naši analizi zmogljivosti. Dodatno smo morali za vse komponente poiskati še tehnična navodila, kar pa pri nekaterih ni bilo mogoče. Osredotočili smo se na komponente za katere smo lahko pridobili vsaj večino potrebnih podatkov.

2.1. MIL 217F standard

Je standard za določevanje intenzivnosti odpovedovanja komponent v vojaški in komercialni industriji, prvotno je bil namenjen samo za vojaške potrebe. MIL-HDBK-217F je priročnik, kjer so zbrani vsi podatki in formule za izračun intenzivnosti odpovedovanja. Standard nam ponuja dve metodi za napovedovanje odpovedi: "parts count prediction" in "parts stress analysis prediction". Odločili smo se za "parts stress analysis prediction", ker imamo napravo, ki ni več v razvoju ampak je že 2 leti v prodaji. Glavna vplivna faktorja, ki jih upoštevamo pri vseh komponentah, sta kvaliteta komponente in okolje delovanja [2.3]. Faktor kvalitete komponente je označen z π_Q, faktor vpliva okolja pa z π_E. Ker je ta standard iz leta 1991, nima podatkov za veliko število komponent, ki se danes uporabljajo pri izdelavi sodobne elektronske opreme. To nam je otežilo analizo in poslabšalo natančnost končne napovedi zanesljivosti naprave.

2.1.1. Enačbe za izračun intenzivnosti odpovedovanja λ (št. odpovedi/10^6 ur):

Procesor (MOS naprave):	$\lambda_p = (C_1 * \pi_T + C_2 * \pi_E) * \pi_Q * \pi_L$
Flash:	$\lambda_p = (C_1 * \pi_T + C_2 * \pi_E + \lambda_{cyc}) * \pi_Q * \pi_L$
Logična vezja:	$\lambda_p = \lambda_{BD} * \pi_{MTG} * \pi_T * \pi_{CD} + \lambda_{BP} * \pi_E * \pi_Q * \pi_{PT} + \lambda_{EOS}$
Upori:	$\lambda_p = \lambda_b * \pi_T * \pi_P * \pi_S * \pi_Q * \pi_E$
Kondenzatorji:	$\lambda_p = \lambda_b * \pi_T * \pi_C * \pi_V * \pi_{SR} * \pi_Q * \pi_E$
Diode:	$\lambda_p = \lambda_b * \pi_T * \pi_S * \pi_C * \pi_Q * \pi_E$
Tranzistorji (MOSFET):	$\lambda_p = \lambda_b * \pi_T * \pi_A * \pi_E * \pi_Q$

Tabela 2.1: Enačbe za izračun intenzivnosti odpovedovanja [2.1]

C_1.....	faktor kompleksnosti vezja	π_{CD}....	korekcijski faktor glede na kompleks. integriranega vezja
C_2.....	faktor ohišja	λ_{BP}....	osnovno odpovedovanje ohišja glede na število nožic
π_T.....	temperaturni faktor	π_{PT}....	korekcijski faktor tipa ohišja
π_E.....	faktor vpliva okolja	λ_{EOS}...	korekcijski faktor za izpostavljenost ESD vplivom
π_Q.....	faktor kvalitete komponente	λ_b.....	osnovna intenzivnost odpovedovanja glede na tip diode
π_L.....	faktor učenja	π_P.....	korekcijski faktor disipacijske moči upora
λ_{CYC}....	faktor bralno/pisalnih ciklov	π_S.....	električni stress faktor
λ_{BD}.....	osnovno odpovedovanje vezja	π_C.....	korekcijski faktor kapacitivnosti kondenzatorja
λ_{MTG}...	korekcijski faktor za proces izdelave	π_A.....	korekcijski faktor namena uporabe tranzistorja

2.1.2. Relex

Za izračun zanesljivosti elektronskih komponent smo uporabili programsko orodje Relex [2.4]. Ker Relex pridobiva podatke iz standarda, je v sam program potrebno vnesti, oziroma izbrati, osnovne podatke o komponentah. Nato program na osnovi teh podatkov izračuna intenzivnost odpovedovanja za vsako komponento. Skupna intenzivnost odpovedovanja je seštevek vseh posameznih intenzivnosti odpovedovanja. Ker je MIL 217F dokaj star standard nima podatkov o večini komponent, ki se danes uporabljajo. Posledično tudi program ne pridobi teh podatkov. V takem primeru smo morali intenzivnost odpovedovanja vpisati ročno.

Mikroprocesor

Samsung Galaxy Tab poganja Samsungov S5PC110A01 procesor [2.5], uporablja 32bitno ARM Cortex A8 jedro, frekvenca delovanja je 1GHz, izdelan v CMOS 45nm tehnologiji in ima vgrajeno POWERVR SGX 3D grafiko. Izvedba tega procesorja je nekoliko kompleksnejša saj so uporabili Package-on-Package (PoP) princip in so na isti čip "nalepili" še dodaten čip Samsung KB100D100YM, ki vsebuje RAM. Zapakiran je v FCFBGA ohišje z 580 nožicami. V Relex smo vnesli tehnologijo izdelave, uporabljeno arhitekturo, št. nožic, kako dolgo je na tržišču in tip ohišja na osnovi katerega nam je izračunal temperaturne koeficiente.

Flash pomnilnik

Za shranjevanje podatkov uporablja 16GB NAND Flash, SanDisk SDIN4C2-16G [2.6], izdelan v NMOS tehnologiji in zapakiran v BGA ohišje z 169 nožicami. V Relex smo vnesli tehnologijo izdelave, št. nožic, velikostni razred (GB), kako dolgo je na tržišču in tip ohišja na osnovi katerega nam je izračunal temperaturne koeficiente.

Diode, kondenzatorji, upori in tranzistorji

Glede na spisek elementov [2.2] katerega smo poiskali na internetu je število diod 27, keramičnih kondenzatorjev 379, SMD "Flat Chip" uporov 195, SMD "Precision" uporov 9 , 5 MOSFET in 2 Bipolarna tranzistorja.

V Relex smo vnesli:

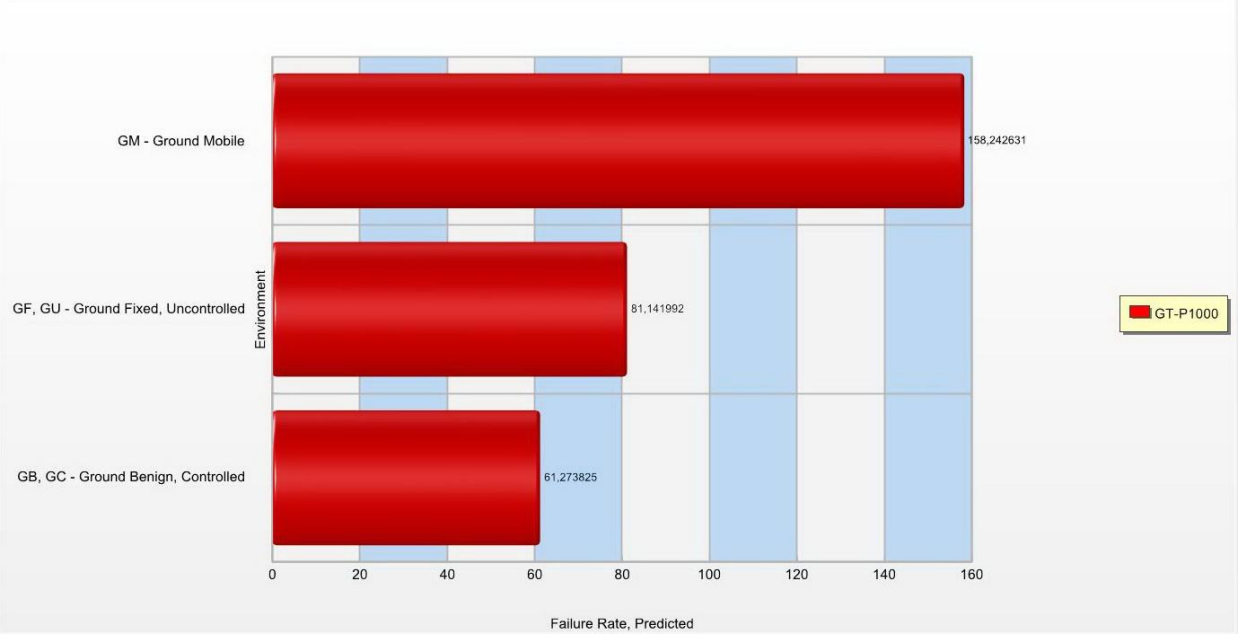
- Diode: nivo kvalitete, tip diode, tip izdelave in temperaturno območje
- Kondenzatorji: nivo kvalitete, velikostni razred, nazivna napetost in temperaturno območje
- SMD "Flat Chip" upori: intenzivnost odpovedovanja smo ročno vnesli, privzeli smo 0,0025
- SMD "Precision" upori: nivo kvalitete, nazivno moč, nazivno napetost in temp. območje

Ostala integrirana vezja

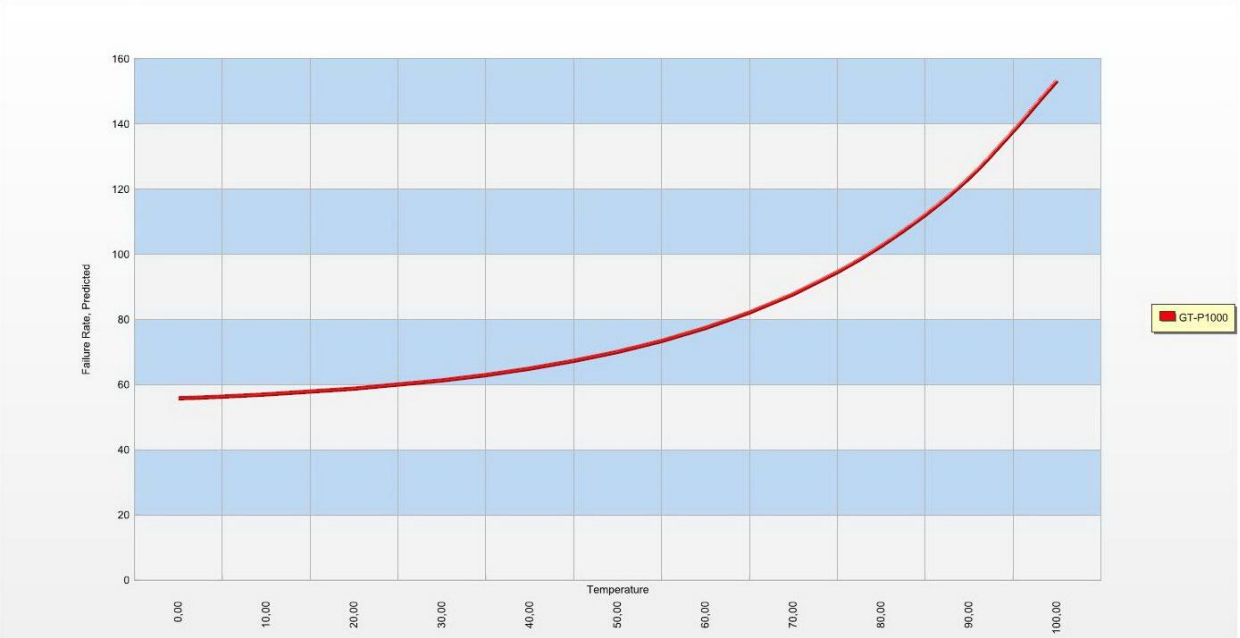
Samsung Galaxy TAB ima še en mikroprocesor Infineon PM9801 [2.7] (uporablja 32bitno ARM 1176 arhitekturo), ki je uporabljen pri rešitvi "triple band" HSxPA in EDGE "quad band" modema z ostalimi integriranimi vezji: Infineon PMB5703 (RF oddajnik), WolfsonMicroelectronics WM8994 (avdio kodek). Tukaj so še Broadcom BCM4329 (Bluetooth/FM/WLAN sprejemnik in oddajnik), STMicroelectronics L3G4200D (digitalni 3-D MEMS Gyroscope), ATMEL MXT224 (krmilnik za ekran na dotik), Broadcom BCM4751 (GPS sprejemnik). Vsa obravnavana integrirana vezja so izdelana v MOS tehnologiji in so zapakirana v BGA ohišja razlikujejo se v številu nožic. V Relex smo vnesli tehnologijo izdelave, uporabljeno arhitekturo, št. nožic, kako dolgo je na tržišču in tip ohišja na osnovi katerega nam je izračunal temperaturne koeficiente.

Name	Manufacturer	PartNumber	Category	Subcategory	Quantity	FailureRate	MTBF
Galaxy TAB	Samsung	GT-P1000			1	61,273825	16320
Microprocessor	Samsung	S5PC110A01	Integrated Circuit	Microprocessor	1	2,116596	472457
Flash	Sandisk	SDIN4C2-16G	Integrated Circuit	Memory	1	0,366652	3000000
Digital 3-D MEMS Gyroscope	STMicroelectronics	L3G4200D	Integrated Circuit	PAL, PLA	1	0,029027	30000000
Bluetooth/FM/WLAN	Broadcom	BCM4329	Integrated Circuit	PAL, PLA	1	0,130256	8000000
RF Transceiver	Infineon	PMB 5703	Integrated Circuit	PAL, PLA	1	0,249684	4000000
Multi-Channel CODEC	Wolfson Microelectronics	WM8994	Integrated Circuit	PAL, PLA	1	0,142986	7000000
X-GOLD 616, HSDPA/HSUPA/EDGE Modem Solution	Infineon	PBM 9801	Integrated Circuit	Microprocessor	1	1,278621	782093
Integrated SP8T switch	TriQuint Semiconductor	TQM6M9014	Integrated Circuit	PAL, PLA	1	0,048303	20000000
Integrated Duplexer	TriQuint Semiconductor	TQM626028L	Integrated Circuit	PAL, PLA	1	0,029027	30000000
Integrated Duplexer	TriQuint Semiconductor	TQM676021	Integrated Circuit	PAL, PLA	1	0,029027	30000000
Integrated Duplexer	TriQuint Semiconductor	TQM666022	Integrated Circuit	PAL, PLA	1	0,029027	30000000
GPS receiver	Broadcom	BCM4751	Integrated Circuit	PAL, PLA	1	0,080358	10000000
Touchscreen controller	Atmel	MXT224	Integrated Circuit	PAL, PLA	1	0,094721	10000000
LVDS Transmitter	Texas Instrument	SN75LVDS83BZQLR	Integrated Circuit	PAL, PLA	1	0,10925	9000000
Resistor Precision SMD			Resistor	Precision,	9	1,545042	647232
Resistor SMD Flat Chip			Resistor	Surface Mount	195	0,4875	2000000
Capacitor Ceramic Multilayer - X5R/X7R			Capacitor	Chip, Ceramic (CDR)	279	4,999817	200007
Capacitor Ceramic Multilayer - C0G/NP0			Capacitor	Chip, Ceramic (CDR)	100	0,782258	1000000
Capacitor	NEC	ESVJ1A335M	Capacitor	Solid, Elec, Tant (CSR)	1	0,020772	50000000
Transistor MOSFET - Dual Complementary 30V/1.5A	Rohm	US6M2	Semiconductor	Transistor	2	0,009829	100000000
Transistor Bipolar - NPN, 50V, 100mA	Rohm	DTC144EM	Semiconductor	Transistor	2	0,007606	100000000
Transistor MOSFET - P-Channel, -20V, -18A	Fairchild Semiconductor	FDMC510P	Semiconductor	Transistor	3	0,001576	600000000
Diode ESD Protect - TVS, Uni-Directional		ESDALC6V1-1M2	Semiconductor	Diode	2	0,01697	60000000
Diode ESD Protect - TVS, 2-Line, Low Capacitance	Semtech	RCLAMP0502B	Semiconductor	Diode	9	0,076363	10000000
Diode ESD Protect - TVS, 1-Line	Semtech	UCLAMP0501T	Semiconductor	Diode	16	0,135757	7000000
Display - 7'' Capative, LED Backlit, 1024x600, 169bpi	Samsung	LMS700JF03	Miscellaneous	Display	1	17,6	56818
Battery - Li-Ion, 3.7V, 4000mAh, 14.8Wh	Samsung	SP4960C3A	Miscellaneous	Battery	1	30,4568	32833
Loudspeaker		BJD5D10915	Miscellaneous	Loudspeaker	2	0,4	3000000

Tabela 2.2: Tabela rezultatov



Slika 2.1: Intenzivnost odpovedovanja glede na okolje delovanja



Slika 2.2: Intenzivnost odpovedovanja glede na temperaturo

2.2. IEC TR 62380

Po standardu IEC TR 62380, smo izračunali intenzivnost odpovedovanja naslednjih komponent:

- mikroprocesor
- FLASH pomnilnik
- prikazovalnik
- baterija
- SMD upor
- keramični kondenzator

2.2.1. Mikroprocesor in FLASH pomnilnik

$$\lambda = \left(\underbrace{\left\{ \lambda_1 \times N \times e^{-0.35 \times a} + \lambda_2 \right\} \times \left\{ \frac{\sum_{i=1}^{y} (\pi_t)_i \times \tau_i}{\tau_{on} + \tau_{off}} \right\}}_{\lambda_{die}} + \underbrace{\left\{ 2.75 \times 10^{-3} \times \pi_\alpha \times \left(\sum_{i=1}^{z} (\pi_n)_i \times (\Delta T_i)^{0.68} \right) \times \lambda_3 \right\}}_{\lambda_{package}} + \left\{ \underbrace{\pi_I \times \lambda_{EOS}}_{\lambda_{overstress}} \right\} \right) \times 10^{-9} / h$$

Slika 2.3: Matematični model za integrirana vezja [2.8]

Parametri, ki jih potrebujemo za izračun:

- $(t_{ae})_i$ … povprečna zunanja ambienta temperatura, ki obkroža opremo, skozi i-to fazo misijskega profila
- $(t_{ac})_i$ … povprečna ambientna temperatura tiskanega vezja blizu komponent, kjer je temperaturni gradient izenačen
- λ_1 … osnovna stopnja odpovedi na tranzistor, glede na družino integriranega vezja
- λ_2 … stopnja odpovedi povezana s tehnologijo izdelave integriranega vezja
- N … število tranzistorjev
- a … (leto izdelave) - 1998
- $(\pi_t)_i$ … i-ti temperaturni faktor povezan z i-to temperaturo stičišča integriranega vezja v misijskem profilu
- τ_i … i-to delujoče časovno razmerje integriranega vezja za i-to temperaturo stičišča v misijskem profilu
- τ_{on} … celotno delujoče časovno razmerje integriranega vezja
- τ_{off} … časovno razmerje integriranega vezja, ko je to v stanju mirovanja
- π_α … faktor vpliva, povezan s spremembo koeficientov toplotnega raztezanja med materialom ohišja in podlago, na kateri se nahaja integrirano vezje
- $(\pi_n)_i$ … i-ti faktor vpliva, povezan z letnim številom temperaturnih sprememb, ki jih integrirano vezje občuti z amplitudo ΔT_i
- ΔT_i … i-ta sprememba temperaturne amplitude v misijskem profilu
- λ_3 … osnovna stopnja odpovedi integriranega vezja
- π_I … faktor vpliva, povezan z uporabo integriranega vezja
- λ_{EOS} … faktor odpovedovanja, povezan z električno preobremenitvijo v upoštevanih aplikacijah

Pri računanju intenzivnosti odpovedovanja (matematični model na sliki 2.1) za integrirana vezja (IC), je pomembno poznati predvsem št. tranzistorjev v čipu, št. nožic, leto izdelave IC in v kakšnem okolju se omenjene komponente uporablja. To so podatki, ki jih moramo sami poiskati, ostali podatki so že navedeni v tabelah in je potrebno poiskati samo ustrezne konstante.

Predvsem se težko dobi podatke o št. tranzistorjev na čip, saj so proizvajalci s temi podatki zelo skopi. Problem pri našem mikroprocesorju je, da imamo v enem ohišju mikroprocesor, RAM ter še nekaj pomnilnika skupaj. Medtem ko smo podatke o št. tranzistorjev za mikroprocesor našli ($200 * 10^6$), podatkov o št. tranzistorjev za pomnilni del pa nismo našli. Tako smo naredili predpostavko, da se za 4GB pomnilnika porabi en tranzistor / bit informacije in za preostalih 8GB en tranzistor/2bit-a informacije (MLC tehnologija izdelave), kar nam potem znese skupaj $64 * 10^9$ tranzistorjev (12GB je skupna velikost vsega pomnilnika na čipu). Tako je skupno število tranzistorjev na čipu $64,2 * 10^9$ in dobimo intenzivnost odpovedovanja 2425,2 FIT ali 47,07 let.

Podobno predpostavko smo naredili tudi v primeru dodatnega FLASH pomnilnika, ki je velik 16GB (MLC tehnologija izdelave) in bi potem vseboval $64,2 * 10^9$ tranzistorjev. Intenzivnost odpovedovanja bi bila potem 254,52 FIT ali 448,51 let.

2.2.2. Prikazovalnik

$$\lambda = \lambda_0 \times \left(1 + 2.5 \times 10^{-2} \times \left[\sum_{i=1}^{j} \left(\pi_n \right)_i \times \left(\Delta T_i \right)^{0.68} \right] \right) \times 10^{-9} / h$$

Slika 2.4: Matematični model za prikazovalnike [2.8]

Parametri, ki jih potrebujemo za izračun:

- λ_0 … intenzivnost odpovedovanja, za določen tip prikazovalnika
- $(\pi_n)_i$ … i-ti faktor vpliva, povezan z letnim številom temperaturnih sprememb, ki ga prikazovalnik občuti z amplitudo ΔT_i
- ΔT_i … i-ta sprememba temperaturne amplitude v misijskem profilu

Pri računanju intenzivnosti odpovedovanja ni potrebno poznati nobenih posebnih podatkov.
Intenzivnost odpovedovanja prikazovalnika je 22039 FIT ali 5,17 let. Edina posebnost pri izračunu je ta, da so v IEC-u opisane konstante za 10 palčni LCD zaslon, Samsung Galaxy TAB pa uporablja 7 palčni zaslon.

2.2.3. Baterija

$$\lambda = \lambda_0 \times 10^{-9} \ / \ h$$

Slika 2.5: Matematični model za baterijo [2.8]

Celoten izračun intenzivnosti odpovedovanja za baterijo, je že opravljen. Vse kar je potrebno vedeti je ali je baterija v napravi primarna ali sekundarna. Baterija v Galaxy Tab-u se uporablja kot primarna in ima zato intenzivnost odpovedovanja 20 FIT. Vendar je pri danem izračunu opomba, ki nas opozori da je pričakovana doba omejene naprave omenjena. Torej moramo sami upoštevati življenjsko dobo baterije, znotraj življenjske dobe pa baterija odpoveduje z intenzivnostjo 20 FIT. Pričakovana življenjska doba Li-Ion baterij je 3-4 leta [2.9] ob pravilni uporabi (polnjenje in praznjenje). Če sedaj sami izračunamo koliko FIT je 3 leta dobe, dobimo 38051 FIT. Na to dodamo še 20 FIT in dobimo da je celotna intenzivnost odpovedovanja baterije 38071 FIT ali 2,99 let.

2.2.4. SMD upor

$$\lambda = 0.01 \times \left(\left[\frac{\sum_{i=1}^{y} (\pi_t)_i \times \tau_i}{\tau_{on} + \tau_{off}} \right] \times \sqrt{N} + 3.3 \times 10^{-3} \times \left[\sum_{i=1}^{j} (\pi_n)_i \times (\Delta T_i)^{0.68} \right] \right) \times 10^{-9} \ / \ h$$

Slika 2.6: Matematični model za SMD upor [2.8]

Parametri, ki jih potrebujemo za izračun:

- glej razdelek 2.2.1
- N … št. uporov v uporovnem nizu (v našem primeru je N=1)
- dejanska operativna moč na uporu
- nazivna moč upora

Pri izračunu za upor je pomembno da poznamo kakšna je nazivna in operativna moč upora. Nazivno moč upora se dobi iz specifikacij medtem, ko podatka o operativni moči ne moremo pridobiti saj ne vemo natanko kakšna je njihova upornost in pri katerih napetostih delujejo. Zato smo naredili predpostavko, da se na uporih uporablja 3.7V (napetost baterije) in da imajo upori dokaj veliko upornost (10K Ohm), saj višja ko je upornost manj toka teče skozi njih in s tem porabijo tudi manj moči. Če uporabimo omenjeno predpostavko dobimo intenzivnost odpovedovanja 0,02399 FIT za en upor. Ker pa imamo podatek o številu uporov v vezju, tj. 204, dobimo da je intenzivnost odpovedovanja vseh uporov 4,894 FIT ali 23326 let.

2.2.5. Keramični kondenzator

$$\lambda = 0.05 \times \left(\left[\frac{\sum_{i=1}^{y} (\pi_t)_i \times \tau_i}{\tau_{on} + \tau_{off}} \right] + 3.3 \times 10^{-3} \times \left[\sum_{i=1}^{j} (\pi_n)_i \times (\Delta T_i)^{0.68} \right] \right) \times 10^{-9} / h$$

Slika 2.7: Matematični model za keramični kondenzator [2.8]

Parametri, ki jih potrebujemo za izračun:

- glej razdelek 2.2.1

Pri izračunu intenzivnosti odpovedovanja keramičnega kondenzatorja, moramo vedeti kakšna je ambientna temperatura. Za izračun smo uporabili temperaturo 30°C in tako dobimo intenzivnost odpovedovanja 0,05348 FIT, kar znese 20,269 FIT ali 5632 let za vseh 379 kondenzatorjev. Vendar je pri izračunu podano opozorilo, kjer je navedeno da mora biti razmerje med maksimalno napetostjo (peak voltage) in nazivno napetostjo (rated voltage) manjše ali enako 0.5 zato, da so rezultati skladni z podanim matematičnim modelom. V našem izračunu smo predpostavili, da je omenjeni pogoj izpolnjen.

2.3. Ugotovitve

Z analizo po MIL-HDBK-217F smo dobili intenzivnost odpovedovanja celotnega sistema 61,27 odpovedi na 10^6 ur, kar znese približno 1,86 let. Ker je večina komponent nepopravljivih, sistem v celoti odpove, so pa tudi komponente, ki so zamenljive in ne vodijo v trajno odpoved sistema. Taki dve komponenti sta baterija in zaslon, ki v največji meri prispevata k večji intenzivnosti odpovedovanja celotnega sistema. Baterija nam odpove že po približno 3,7 letih in zaslon po približno 6,5 letih. Naslednja izmed komponent, ki povzroči trajno odpoved sistema, je keramični kondenzator oz. skupina keramičnih kondenzatorjev, ki jim je Relex skupaj napovedal 22,81 let delovanja.

Rezultate dobljene po MIL standardu lahko primerjamo z novejšim IEC standardom. Največje odstopanje opazimo pri intenzivnosti odpovedovanja za kondenzatorje in upore. IEC napoveduje odpovedovanje teh komponent v tisoč letih, medtem ko MIL napoveduje v desetletjih. Do takih razlik prihaja verjetno zaradi tega, ker je pri računanju po IEC standardu potrebno v izračune vnesti veliko več podrobnosti, ki se jih pa zelo stežka poišče oz. se jih ne da pridobiti. Zato si rezultate dobljene po IEC standardu interpretiramo kot oceno in ne kot točnih napovedi, ker je bilo pri izračunih narejenih kar nekaj predpostavk, ki pa lahko močno vplivajo na končni rezultat.

3. Analiza programske opreme (Cesar, Sakelšak, Škaper)

3.1. Operacijski sistem Android

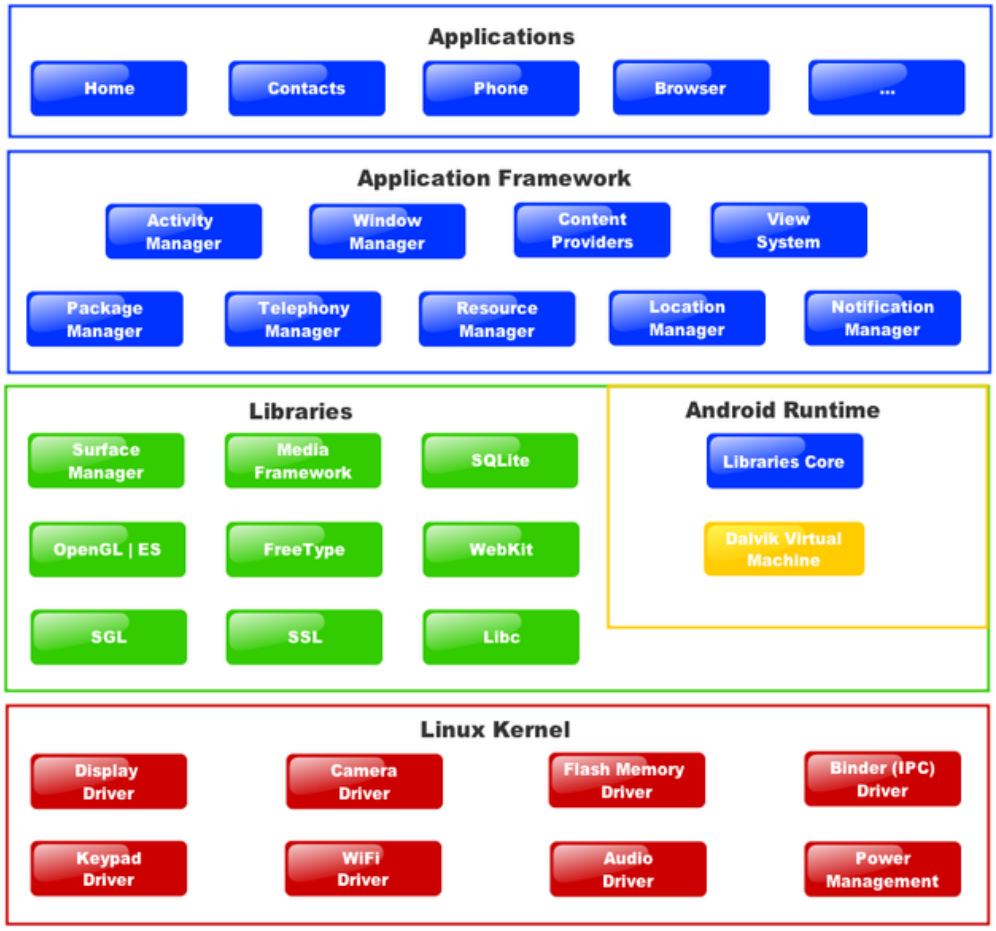
Android je programska platforma in operacijski sistem za pametne mobilne telefone, ki temelji na Linux jedru. Razvija ga Google v sodelovanju s podjetji združenja Open Handset Alliance (OHA). Android je v prvi vrsti namenjen mobilnim telefonom, v prihodnosti pa naj bi z njim opremili tudi nekatere druge naprave. Android je na voljo kot open source, ki je izdan pod odprto-kodno Apache licenco, dodatki za Linux kernel pa pod GPL v2 licenco. Vsa izvorna koda je dostopna preko repozitorijev, kjer se izvorna koda tudi posodablja. Od prve različice operacijskega sistema Android je bilo izdanih že kar nekaj različic. Izdane različice so podane v tabeli:

Različica	Kodno ime	Različica jedra	Datum izida
1.5	Cupcake	2.6.27	30 April 2009
1.6	Conut	2.6.29	15 September 2009
2.0/2.1	Eclair	2.6.29	26 October 2009
2.2	Froyo	2.6.32	20 May 2010
2.3	Gingerbread	2.6.35	6 December 2010
3.0	Honeycomb	2.6.36	22 Februar 2011
V pripravi	Ice Cream Sandwich	V pripravi	V pripravi

Tabela 3.1: Različice operacijskega sistema Android

Različici 1.5 in 1.6 sta bili le nadgradnji začetne različice, medtem ko je različica 2.0/2.1 prinesla novosti na področju podpore multi-touch zaslonov, podpore poljubnim resolucijam zaslonov brez dodatnega prilagajanja aplikacij, naprednejše podpore navidezni tipkovnici, podpore HTML5 standardu v vgrajenem brskalniku ipd. Z različico 2.2 so prišle glavne spremembe, ki vključujejo izboljšano hitrost delovanja z JIT(just in time) optimizacijo, integracijo Chrome V8 JavaScript engine v spletni brskalnik, funkcionalnost Wi-Fi hotspot in podporo Adobe Flash-a. Različica 2.3 je prinesla izboljšave na področju uporabniškega vmesnika, izboljšane programske tipkovnice in funkcionalnosti copy/paste. Tabelno-orientirano različico smo dobili s prihodom izdaje 3.0, ki podpira večje zaslone naprav in predstavlja mnogo novih funkcionalnosti uporabniškega vmesnika. Poleg tega podpira tudi več jedrne procesorje in strojno pospeševanje za grafiko. Prihajajoča verzija Ice Cream Sandwich je kombinacija različice 2.3 in 3.0, ki bo združila dosedanje različice za telefone z različico 3.0 namenjeno izključno tabličnim računalnikom. Izšla naj bi predvidoma sredi leta 2011.

3.2. Arhitektura androida



Slika 3.1: Arhitektura Android OS (Vir: http://www.cellphoneanswers.info/android-architecture/)

Diagram prikazuje večino komponent operacijskega sistema Android. Operacijski sistem Android je sestavljen iz petih delov, in sicer:

- Applications
  Na aplikacijskem nivoju se nahaja množica aplikacij, kot so email klient, SMS program, koledar, iskalnik, kontakti, in druge. Vse aplikacije so pisane v programskem jeziku Java.

- Application Framework
  Z aplikacijskim ogrodjem, ki je objektno orientiran razvijalci zgradijo bogate in inovativne aplikacije.

- Libraries
  Android vključuje množico C/C++ knjižnic, ki jih uporabljajo različne komponente operacijskega sistema Android. Te uporabljajo razvijalci skozi aplikacijsko ogrodje Androida ali neposredno preko NDK (Native Development Kit) okolja.

Nekatere knjižnice:

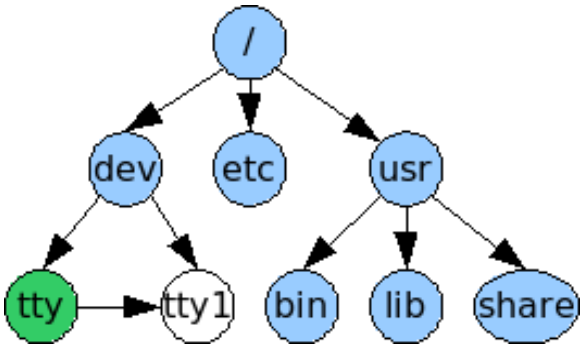
- System C library (standardna C sistemska knjižnica(libc))
- Media Libraries (knjižnica podpira mnogo popularnih audio in video formatov(MPEG4, H.264, MP3, AAC, AMR, JPG in PNG))
- Surface Manager(upravlja dostop do podsistema za prikaz na zaslonu)
- LibWebCore(sodoben spletni brskalnik, ki temelji na Webkit pogonu)
- SQLite(močna in lahka relacijska zbirka podatkov, dostopna vsem aplikacijam)

- Android Runtime
  Android vključuje množico knjižnic, ki zagotavljajo večino funkcionalnosti, ki je dostopna v knjižnicah programskega jezika Java. Vsaka Android aplikacija teče na svojem procesu, s svojo lastno instanco virtualne naprave Dalvik. Dalvik je bil napisan tako, da lahko na napravi efektivno teče več virtualnih naprav. Virtualna naprava Dalvik se opira na funkcionalnost Linux jedra kot je nitenje in nizko nivojsko upravljanje s pomnilnikom.

- Linux Jedro
  Android 2.2 uporablja Linux jedro različice 2.6.32, z dodatki za naprave na katerih teče. Pod te dodatke spadajo gonilniki za module, ki jih najdemo na telefonih in ne na klasičnih računalnikih, ki so primarna platforma Linux jedra. Gonilniki so za Bluetooth modul, za Wi-Fi modul, za GSM/3g anteno in podobne. Samo Linux jedro v android izvorno kodo ni vključeno, pač pa tam najdemo zgolj omenjene dodatke in popravke (t.i. patche) za jedro. Omenjeni dodatki so bili na začetku tudi v repozitoriju izvorne kode Linux jedra, vendar so jih zaradi licenčnih konfliktov z Googlom odstranili, kar efektivno pomeni, da gre pri Androidu za povsem svojo različico Linux jedra.

3.2.1. Datotečni sistem

Ker je jedro Android operacijskega sistema Linux jedro in ima vgrajeno podporo velikemu številu datotečnih sistemov, bi v osnovi lahko izbirali med njimi, a je zaradi varčevanja s prostorom je veliko teh izklopljenih.
Za primarno sistemsko particijo uporablja datotečni sistem YAFFS2, ki je optimiziran za flash pomnilniške kartice, ker omogoča razporejanje pisanj čez celotno površino pomnilnika, saj imajo flash pomnilniški bloki številno mnogo količino pisanj vanje. Poleg YAFFS2, je podprt datotečni sistem še FAT32, ki se je ustalil kot de-facto standard pomnilniških kartic in USB ključev zaradi podpore inferiornim operacijskim sistemom. Kot zanimivost omenimo da je Android z različico 2.3 začel uporabljati popularen datotečni sistem ext4.
Linux ima urejen navidezni datotečni sistem po POSIX standardu kar s seboj pripelje strukturo imenikov:



Slika 3.2: Datotečna struktura POSIX standarda (Vir:http://www.jamesmolloy.co.uk/tutorial_html/8.-The%20VFS%20and%20the%20initrd.html)

3.2.2. Zgradba projekta

Izvorno kodo pridobimo z uporabno Git odjemalca ter spletnih repozitorijev, ki so na voljo na naslovu: http://android.git.kernel.org/

Modul	Opis
bionic	Osnovne sistemske knjižnice in orodja
bootable	Zagonski nalagalnik, orodja za ustvarjanje zagonskih diskov
build	Sistem za prevajanje projekta - "Build system"
cts	Testno okolje za kompatibilnost
dalvik	Navidezni računalnik za izvajanje javanskih programov
development	Orodja za razvoj platforme in prototipska koda
device	Datoteke in gonilniki za specifične naprave
external	Potrebna sistemska orodja, ki pa niso del projekta
frameworks	Ogrodja za razvoj aplikacij za različne raličice
hardware	Programje za abstrakcijo strojne opreme
ndk	Vmesnik za povezovanje Android (Java) programja s C/C++ knjižnicami
packages	Osnovni Android programi
prebuilt	Binarne datoteke, za podporo prevajanju
sdk	Koda razvojnega okolja
system	Sistemska orodja za podporo protokolom

Tabela 3.2: Datotečna struktura pridobljenega projekta

Linux jedro predstavlja približno polovica izvorne kode v jeziku C, preostala polovica so knjižnice in sistemska programska oprema, ki jedro "uporabljajo". Javanskih datotek je približno 18000, kar je več kakor v prejšnji različici, saj omogoča veliko novih funkcionalnosti.

3.3. Statična analiza programske opreme

3.3.1. Način testiranja

Izvorno kodo smo testirali s pomočjo programske opreme podjetja Testwell. Uporabili smo orodje CMT++ za merjenje kompleksnosti izvorne kode v programskem jeziku C/C++ in orodje CMTJava za merjenje kompleksnosti izvorne kode v jeziku Java. Zaradi problema pri pridobitvi grafičnega orodja Verybench, ki združuje oba jezika, smo predstavili rezultate samo v tekstovnem formatu.

Poleg celotnega operacijskega sistema Android verzije 2.2 smo testirali tudi Linux jedro verzije 2.6.32. in uporabniške aplikacije, katere so bile že testirane v lanskoletnem poročilu, da smo primerjali kako so aplikacije napredovale v enem letu.

3.3.2. Metrike kompleksnosti programske opreme

Za ocenjevanje zanesljivosti in kompleksnosti izvorne kode programov obstajata dva deterministična modela, in sicer McCabe-jeva in Halstead-ova mera. McCabe-jeva mera, ki jo imenujemo tudi ciklomatična kompleksnost, nam poda oceno o kompleksnosti vejitev v programu. S Halstead-ovo mero ugotovimo število napak v programu. Testwellova orodja za testiranje programske opreme uporabljajo metrike iz obeh modelov in sicer:

- Ciklomatično število v(G);
- Število vrstic programske kode LOCpro;
- Delež komentarjev;
- Volumen programa V;
- Predvideno število programskih napak;
- Indeks zahtevnosti vzdrževanja MI.

McCabe-jevo ciklomatično število v(G)

McCabe-jevo ciklomatično število v(G) nam predstavlja število neodvisnih poti v programu, hkrati pa poda tudi zgornjo mejo števila testov, ki so potrebni da se vsak ukaz v programu izvede vsaj enkrat. Z večjim številom testov se pojavi tudi več poti skozi program, kar privede do težje razumljivosti programa. McCabe-jevo ciklomatično število v(G) je neodvisno od podatkovnih struktur in njihove kompleksnosti. Statistike kažejo, da je zgornja meja v(G) enaka 10, ker je večje število poti težko dobro testirati. Dovoljene meje, ki jih navaja proizvajalec programskega orodja, so od 1 do 15. Število v(G) povečujejo **if** stavki, **for** in **while** zanke, vsak **case** stavek stavka **switch** (razen default) in sestavljeni pogoji tipa **&&** ali **||**.

Halstead-ova mera

Halstead-ovo mero sestavljata volumen programa in predvideno število napak v programu. Oba podatka se izračunata iz osnovnih štirih, kot so:

- število različnih operatorjev(n1);
- število različnih operandov(n2);
- število vseh operatorjev(N1)
- število vseh operandov(N2)

Operandi so definirani kot:

- spremenljivke s podatkovnimi tipi(*intiger*, *double*, *boolean*,...);
- ključne besede(*false*, *true*, super, *void*, *this*);

- vse druge oznake, ki niso ključne besede;
- literali (numerični literali in literali iz izrazov).

Vso ostalo izvorno kodo se obravnava kot operatorje. Operatorje predstavljajo na primer stavki **if**, **for**, **case**, itd. Vse vrste oklepajev ((…), […],{...}), ki nastopajo v parih se obravnavajo kot en operator.

Do podatkov o volumnu programa in številu napak programske kode pridemo s pomočjo formul in sicer:

- Volumen programa dobimo iz vmesnih podatkov s pomočjo formule **V = N * log2(n)**, kjer je N = LOCpro = N1+ N2, n pa velikost slovarja;
  Vmesni podatki:
  - N-dolžina programa: N=N1+N2 (vsota vseh operandov in operatorjev);
  - n-velikost slovarja: n=n1+n2 (vsota vseh različnih operatorjev in operandov).

- Predvideno število napak v programu se izračuna iz vmesnih podatkov po formuli **B=(E^(2/3))/3000**, kjer se dobljena vrednost pomnoži s korelacijskim faktorjem, ki je določen v konfiguracijski datoteki.
  Vmesni podatki:
  - D - nivo težavnosti: D=(n1/N1)*(N2/n2) (sorazmerno številu vseh operatorjev in vseh različnih operandov);
  - L - nivo programa: L=1/D (inverz nivoja težavnosti, program z nizkim nivojem bolj občutljiv, kot program z visokim nivojem);
  - E - težavnost implementacije: E = V*D (sorazmeren volumnu in nivoju težavnosti);
  - T - čas za implementacijo oz. čas za razumevanje programa: T=E/18 (zadostna aproksimacija časa v sekundah - ugotovil Halstead)

Priporočene mejne vrednosti s strani proizvajalca programske opreme so od 0 do 2 predvideni napaki.

Indeks zahtevnosti vzdrževanja

Indeks zahtevnosti vzdrževanja zavzema vrednosti pod 100 in podaja oceno kakšna je težavnost vzdrževanja projekta. S spremljanjem indeksa MI lahko preprečimo ali zmanjšamo entropijo izvorne kode in tako povečamo integriteto. Indeks MI nam pove tudi kdaj se splača napisati nov program namesto popravljanja in dodajanja novih funkcionalnosti v obstoječi program.

Indeks možnosti vzdrževanja MI se računa za vsako funkcijo, razred ali strukturo in datoteko posebej, ter za vse datoteke skupaj. Računamo ga po formuli **MI=MIwoc+Micw**, kjer MIwoc predstavlja MI brez komentarjev, Micw pa predstavlja utež za komentarje. Miwoc in Micw izračunamo s pomočjo formul:

- Miwoc=171-5.2*ln(aveV)-0.23*aveG-16.2*ln(aveLOC), kjer vmesni rezultati aveV, aveG in aveLOC pomenijo naslednje:
  - aveV(povprečna velikost slovarja(V) na element testiranja*);
  - aveG(povprečna ciklomatična kompleksnost v(G) na element testiranja*);
  - aveLOC(povprečno število vrstic(LOCphy) na element testiranja*).

- Micw=50*sin(sqrt(2.4*perCM)), kjer vmesni rezultat perCM pomeni povprečni odstotek vrstic s komentarji na element testiranja*

Element testiranja* pomeni v našem primeru funkcijo, razred, strukturo ali datoteko.

Pomen vrednosti MI:

- več kot 85: lahko vzdrževanje
- med 65 in 85: srednja težavnost vzdrževanja
- manj od 65: težavno vzdrževanje

3.4. Meritve

Rezultate meritev bomo predstavili v obliki tekstovnega izpisa, ki nam ga poda orodje, s katerim testiramo izvorno kodo (CMT++ ali CMTJava). S testiranjem C programske kode dobimo izvleček, kjer so za vsako od šestih mer navedene mejne vrednosti, znotraj katerih se ne prožijo alarmi. V izvlečku se nahaja podatek o številu datotek, ki presežejo mejne vrednosti in prožijo alarm. Enak podatek se nahaja tudi za vsako izmed funkcij. Izvleček vsebuje tudi podatke o številu vseh vrstic v datoteki, številu vrstic s programsko kodo, praznih vrstic, vrstic s komentarji ter število podpičij. Pri razlagi meritev smo se omejili samo na datoteke oziroma v primeru Jave na razrede. Pri testiranju Java izvorne kode z CMTJava orodjem imajo izvlečki drugačno obliko, saj nam orodje izvaja analizo po razredih, ki niso nujno vsak v svoji datoteki, namesto po datotekah. Tako nam orodje na podlagi rezultatov v funkcijah izdela oceno za razred, ker se funkcije v Javi nahajajo v razredih.

3.4.1. Jedro 2.6.32

	13081 Files			224451 Functions		
Measure	Alarmed	%	Limits	Alarmed	%	Limits
Cyclomatic number v(G)	2475	18	1-100	10731	4	1-15
Program lines LOCpro	4633	35	4-400	34067	15	4-40
Comment %	11128	85	30-75	129547	57	30-75
Volume V	7462	57	100-8000	46392	20	20-1000
Estimated number of bugs B	7308	55	0-2	0	0	n/a
Maintainability index MI	211	1	65-	6423	2	65-
Total	33217	42		227160	20	

Tabela 3.3: Rezultati dobljeni pri testiranju jedra

Files: 13081
LOCphy: 8836164
LOCbl: 1222245
LOCpro: 6298785
LOCcom: 1532169
';': 3078982
v(G) : 862816
MI without comments : 80
MI comment weight : 28
MI: 108

Razvijalci so pri verziji Androida 2.2 uporabili Linux jedro 2.6.32 namesto verzije 2.6.29, ki ga je uporabljal predhodnik Androida 2.2. Po testiranju jedra 2.6.32 dobimo rezultate prikazane v zgornji tabeli, kjer so najbolj kritične meritve pri deležu komentarjev, volumnu programa in ocenjenem številu hroščev(napak), saj vrednosti(delež datotek) krepko presežejo postavljene okvirje znotraj katerih bi se morali nahajati rezultati.

Delež datotek s preseženim številom komentarjev je 85%, medtem ko je pri verziji 2.6.29 le ta 65%. Opazimo, da so izmerjeni deleži datotek z alarmom pri verziji 2.6.32 večji kot pri verziji 2.6.29, razen indeksa MI, ki je pri verziji 2.6.32 manjši, kar pomeni, da se je zmanjšala zahtevnost vzdrževanja. Zaskrbljujoč je podatek o deležu ocene števila napak, ki je pri verziji 2.6.32 55%, pri verziji 2.6.29 pa 33%.

3.4.2. Celoten projekt, C/C++ koda brez jedra

	14276 Files			175313 Functions		
Measure	Alarmed	%	Limits	Alarmed	%	Limits
Cyclomatic number v(G)	1990	13	1-100	10619	6	1-15
Program lines LOCpro	3450	24	4-400	32257	18	4-40
Comment %	9264	64	30-75	85850	48	30-75
Volume V	5776	40	100-8000	33119	18	20-1000
Estimated number of bugs B	5554	38	0-2	0	0	n/a
Maintainability index MI	527	3	65-	6176	3	65-
Total	26561	31		168021	19	

Tabela 3.4: Rezultati dobljeni pri testiranju sistemskih knjižnic

Files: 14276
LOCphy: 7068047
LOCbl: 898232
LOCpro: 4932693
LOCcom: 1367085
';': 2272993
v(G) : 77811
7MI without comments : 80
MI comment weight : 28
MI: 108

Operacijski sistem Android 2.2 vsebuje poleg jedra še 14276 datotek izvorne kode v programskem jeziku C ali C++, oziroma okoli 4932693 vrstic programske kode(LOCpro). Delež datotek, ki prožijo alarm je nižji kot pri Linux jedru, zato se rezultati bolj približajo mejnim vrednostim, čeprav je stanje še daleč od idealnega. Največji delež datotek z alarmom tvorijo ponovno komentarji, najmanjši pa indeks MI.

3.4.3. Sistem in knjižnice

V to metriko so vključene vse datoteke direktorijev ‘hardware’, ‘bionic’ in ‘system’.

	1704 Files			18522 Functions		
Measure	Alarmed	%	Limits	Alarmed	%	Limits
Cyclomatic number v(G)	230	13	1-100	1211	6	1-15
Program lines LOCpro	405	23	4-400	3645	19	4-40
Comment %	802	47	30-75	8650	46	30-75
Volume V	673	39	100-8000	3909	21	20-1000
Estimated number of bugs B	635	37	0-2	0	0	n/a
Maintainability index MI	54	3	65-	742	4	65-
Total	2799	27		18157	19	

Tabela 3.5: Rezultati dobljeni pri testiranju sistemskega promja in knjižnic

Files: 1704
LOCphy: 903672
LOCbl: 125021
LOCpro: 584546
LOCcom: 210945
';': 286835
v(G) : 85339
MI without comments : 74
MI comment weight : 31
MI: 106

Sem spadajo knjižnice kot je libdl (knjižnica za dinamično povezovanje - dynamic linking), libc (GNU sistemska knjižnica, ki definira vse osnovne funkcije operacijskega sistema - open, malloc, printf, …), libstdc++ (GNU standardna knjižnica za C++ je zbirka funkcij in klasov, ki so del ISO C++ standarda), libthread (knjižnica za nitenje), linker, libm (knjižnica matematičnih operacij). V metriko so vključeni tudi gonilniki za strojno opremo (naprimer za broadcom, qualcomm, texas instruments in druge) in orodja, namenjena delovanju sistema (za razliko od prej omenjenih knjižnic, ki so namenjene podpori aplikacijam napisanim za Android), kot naprimer fastboot, adb (vmesnik za razhroščevanje), orodja za wlan in bluetooth, …
Skupaj je bilo v test zajetih 1704 datotek, največ alarmov pa je bilo sproženih pri metriki komentarjev in sicer pri 47% datotekah. Glede na skupen MI indeks je vzdrževanje teh knjižnic zelo lahko.

3.4.4. Celoten projekt, Java

OVERALL SUMMARY:

Methods (on two top levels)		Classes		Interfaces	
173312	Total	23068	Total	2036	Total
10	Average LOCpro	16756	On top level	1399	On top level
24	Average LOCcom/LOCphy %	14342	Extends	439	Extends
2	Average v(G)	4765	Implements	1	Implements

Packages		Files	
2086	Total	17929	Total
		3833038	LOCphy
		2257160	LOCpro
System		1147718	LOCcom
		456644	LOCbl
233027	v(G)	1116428	';'
96	MI without comments	213	Average LOCphy
35	MI comment weight	30	Average LOCcom/LOCphy %
130	MI	111355	Java comment blocks

Measure	Measured	Alarmed	%	Limits 1st level class	Limits 2nd level class	Limits method
v(G)	198144	**4628**	**2**	1-100	1-100	1-10
LOCpro	198144	**9104**	**4**	4-400	4-400	1-40
Comment %	198144	**47318**	**23**	20-60	20-60	20-60
V	198144	**19232**	**9**	100-8000	100-8000	4-1000
B	198144	**2734**	**3**	0-2	0-2	n/a
MI	198144	**4882**	**2**	80-	80-	80-
Total	1188864	**87898**	**7**			

Tabela 3.6: Rezultati dobljeni pri testiranju javanskega dela projekta

Pri testiranju izvorne kode v Java programskem jeziku dobimo najboljše rezultate, saj se z meritvami najbolj približamo mejam. Zaradi tega je delež datotek, ki presežejo okvirne meje, nizek, saj je alarm katerekoli vrste sprožilo le 7% datotek. Ciklomatično kompleksnost je preseglo le za 2% vseh datotek, število vrstic programske kode za 4%, delež komentarjev za 23%, volumen programa za 9%, predvideno število napak v programu za 3% in indeks zahtevnosti vzdrževanja za 2%.

Iz teh podatkov opazimo, da je izvorna koda v programskem jeziku Java kvalitetnejša od programske kode C in C++. Sklepamo, da Google uporablja pri razvoju operacijskega sistema Android neke vrste sistem za nadzorovanje kvalitete izvorne kode. Predvidevamo, da pri programskem jeziku Java dobimo boljše rezultate predvsem zaradi manjše kompleksnosti aplikacijskega nivoja, kjer se izvorna koda Java nahaja, medtem ko so sistemski deli (Linux jedro) napisani v C-ju bolj kompleksni.

3.4.5. Aplikacije

Brskalnik

```
OVERALL SUMMARY:

Methods (on two top levels)      Classes                 Interfaces
==============================   ====================    ====================
   829 Total                         94 Total                 3 Total
    12 Average LOCpro                52 On top level          1 On top level
    17 Average LOCcom/LOCphy %       62 Extends               0 Extends
     2 Average v(G)                  23 Implements            0 Implements

Packages                         Files
==============================   ============================================
     3 Total                         53 Total
                                  18986 LOCphy
                                  13285 LOCpro
System                             3913 LOCcom
==============================     1955 LOCbl
  1665 v(G)                        6871 ';'
    94 MI without comments          358 Average LOCphy
    30 MI comment weight             20 Average LOCcom/LOCphy %
   124 MI                           354 Java comment blocks

                                      Limits           Limits          Limits
Measure    Measured Alarmed   %   1st level class  2nd level class    method
================================================================================
v(G)            926      33   3       1-100            1-100            1-10
LOCpro          926      61   6       4-400            4-400            1-40
Comment %       926     261  28       20-60            20-60           20-60
V               926     109  11     100-8000         100-8000           4-1000
B               926      16   2       0-2              0-2              n/a
MI              926      30   3       80-              80-              80-
================================================================================
Total          5556     510   9
```

Tabela 3.7: Rezultati dobljeni pri testiranju brskalnika

Kalkulator

```
OVERALL SUMMARY:

Methods (on two top levels)      Classes                 Interfaces
==============================   ====================    ====================
   114 Total                         15 Total                 0 Total
     7 Average LOCpro                14 On top level          0 On top level
     6 Average LOCcom/LOCphy %       10 Extends               0 Extends
     1 Average v(G)                   2 Implements            0 Implements

Packages                         Files
==============================   ============================================
     2 Total                         14 Total
                                   1680 LOCphy
                                   1125 LOCpro
System                              295 LOCcom
==============================      264 LOCbl
   115 v(G)                         639 ';'
   108 MI without comments          120 Average LOCphy
    20 MI comment weight             17 Average LOCcom/LOCphy %
   127 MI                             8 Java comment blocks

                                      Limits           Limits          Limits
Measure    Measured Alarmed   %   1st level class  2nd level class    method
================================================================================
v(G)            129       2   1       1-100            1-100            1-10
LOCpro          129       0   0       4-400            4-400            1-40
Comment %       129      39  30       20-60            20-60           20-60
V               129       1   0     100-8000         100-8000           4-1000
B               129       0   0       0-2              0-2              n/a
MI              129       0   0       80-              80-              80-
================================================================================
Total           774      42   5
```

Tabela 3.8: Rezultati dobljeni pri testiranju kalkulatorja

Kamera

```
OVERALL SUMMARY:

Methods (on two top levels)      Classes                  Interfaces
==============================   ======================   ======================
   846 Total                        109 Total                11 Total
     9 Average LOCpro                76 On top level          2 On top level
     7 Average LOCcom/LOCphy %       64 Extends               0 Extends
     2 Average v(G)                  33 Implements            0 Implements

Packages                         Files
==============================   ============================================
     5 Total                         76 Total
                                  13654 LOCphy
                                   9759 LOCpro
System                             2074 LOCcom
==============================     1853 LOCbl
  1102 v(G)                        5380 ';'
   103 MI without comments          179 Average LOCphy
    20 MI comment weight             15 Average LOCcom/LOCphy %
   124 MI                            76 Java comment blocks

                                        Limits           Limits           Limits
Measure     Measured Alarmed   %   1st level class  2nd level class     method
================================================================================
v(G)             966       8   0       1-100            1-100            1-10
LOCpro           966      24   2       4-400            4-400            1-40
Comment %        966     335  34       20-60            20-60            20-60
V                966      59   6     100-8000         100-8000            4-1000
B                966      10   2         0-2              0-2              n/a
MI               966       9   0         80-              80-              80-
================================================================================
Total           5796     445   7
```

Tabela 3.9: Rezultati dobljeni pri testiranju programa za zajem kamere

Galerija

```
OVERALL SUMMARY:

Methods (on two top levels)      Classes                  Interfaces
==============================   ======================   ======================
   736 Total                        78 Total                 12 Total
    10 Average LOCpro               54 On top level           3 On top level
     7 Average LOCcom/LOCphy %      42 Extends                0 Extends
     2 Average v(G)                 25 Implements             0 Implements

Packages                         Files
==============================   ============================================
     2 Total                         49 Total
                                  12748 LOCphy
                                   9322 LOCpro
System                             1715 LOCcom
==============================     1755 LOCbl
  1083 v(G)                        4826 ';'
   100 MI without comments          260 Average LOCphy
    21 MI comment weight             13 Average LOCcom/LOCphy %
   121 MI                            63 Java comment blocks

                                        Limits           Limits           Limits
Measure     Measured Alarmed   %   1st level class  2nd level class     method
================================================================================
v(G)             826      18   2       1-100            1-100            1-10
LOCpro           826      38   4       4-400            4-400            1-40
Comment %        826     296  35       20-60            20-60            20-60
V                826      63   7     100-8000         100-8000            4-1000
B                826      13   2         0-2              0-2              n/a
MI               826      26   3         80-              80-              80-
================================================================================
Total           4956     454   9
```

Tabela 3.10: Rezultati dobljeni pri testiranju programa multimedijske galerije

Email

```
OVERALL SUMMARY:

Methods (on two top levels)      Classes                 Interfaces
============================     ====================    ====================
  4533 Total                        511 Total               36 Total
    11 Average LOCpro               348 On top level        23 On top level
    25 Average LOCcom/LOCphy %      334 Extends              3 Extends
     2 Average v(G)                 118 Implements           0 Implements

Packages                         Files
============================     ============================================
    30 Total                        372 Total
                                 103694 LOCphy
                                  62627 LOCpro
System                            31211 LOCcom
============================      10275 LOCbl
  7812 v(G)                       33477 ';'
    94 MI without comments          278 Average LOCphy
    35 MI comment weight             30 Average LOCcom/LOCphy %
   130 MI                          2953 Java comment blocks

                                      Limits          Limits          Limits
Measure     Measured Alarmed   %  1st level class  2nd level class  method
================================================================================
v(G)            5080     198   3       1-100           1-100           1-10
LOCpro          5080     272   5       4-400           4-400           1-40
Comment %       5080    1100  21       20-60           20-60          20-60
V               5080     532  10     100-8000        100-8000          4-1000
B               5080      84   2       0-2             0-2             n/a
MI              5080     121   2       80-             80-            80-
================================================================================
Total          30480    2307   7
```

Tabela 3.11: Rezultati dobljeni pri testiranju programa za elektronsko pošto

Predvajalnik glasbe

```
OVERALL SUMMARY:

Methods (on two top levels)      Classes                 Interfaces
============================     ====================    ====================
   559 Total                         65 Total                6 Total
    14 Average LOCpro                35 On top level         0 On top level
     9 Average LOCcom/LOCphy %       51 Extends              0 Extends
     3 Average v(G)                  17 Implements           0 Implements

Packages                         Files
============================     ============================================
     4 Total                         35 Total
                                  13552 LOCphy
                                  10504 LOCpro
System                             1701 LOCcom
============================       1389 LOCbl
  1431 v(G)                        5823 ';'
    92 MI without comments          387 Average LOCphy
    23 MI comment weight             12 Average LOCcom/LOCphy %
   115 MI                           111 Java comment blocks

                                      Limits          Limits          Limits
Measure     Measured Alarmed   %  1st level class  2nd level class  method
================================================================================
v(G)             628      37   5       1-100           1-100           1-10
LOCpro           628      56   8       4-400           4-400           1-40
Comment %        628     272  43       20-60           20-60          20-60
V                628     103  16     100-8000        100-8000          4-1000
B                628      15   3       0-2             0-2             n/a
MI               628      37   5       80-             80-            80-
================================================================================
Total           3768     520  13
```

Tabela 3.12: Rezultati dobljeni pri testiranju predvajalnika glasbe

Testirali smo tudi nabor uporabniških aplikacij, ki smo jih primerjali z lanskoletno verzijo predhodnika Androida 2.2. Vidimo, da pri nekaterih aplikacijah delež komentarjev ni v predpisanih mejah, drugače pa ni korenitih sprememb. Pri primerjavi s prejšnjimi različicami zasledimo rahlo povečan obseg datotek in vrstic programske kode, kar nakazuje postopno dodajanje funkcionalnosti.

Testiranje delovanja aplikacij

Testiranje delovanja aplikacij se opravlja z uporabo *monkeyrunner* orodja, ki je priložen razvojnemu okolju. Orodje omogoča tako pisanje testov kakor razširitve z uporabo jezika Python ter Jython interpreterja, ki uporablja Java navidezni stroj in s tem omogoča enostavno pisanje testnih skript, ter povezljivost z razvitimi javanskimi aplikacijami.

Monkeyrunner se preko ADB (Android Debug Bridge) poveže na Dalvik navidezni stroj ter simulira pritiske fizičnih gumbov (akcije). S pravim zaporedjem “pritiskov” dosežemo testiranje določenega toka zahtev in s tem ugotavljamo pravilno delovanje aplikacij.

Ker v našem primeru nismo namensko testirali neke specifične aplikacije, smo uporabili *monkeyrunner* z “monkey” obnašanjem, kar namesto nekega določenega zaporedja generira psevdo naključne zahtevke in s tem oponaša opico, ki skače po telefonu.

Test je kot rezultat pokazal, da pri večjemu številu akcij in zelo hitri opici, odpove nekaj programov, ki pa se neopazno ponovno postavijo na svoje mesto in s tem ponovno omogočijo vnaprejšnjo zlorabo aparata. Monkeyrunner šele čez čas, ko je napak preveč, oziroma pride do resnejših, prekine svoje akcije in javi mesta kjer je prišlo do napak.

Poskusili smo tudi delovanje po določenem zaporedju akcij z uporabo sledeče skripte:

```
for i in range(1, 1000):
        print i
        device.press('KEYCODE_DPAD_DOWN','DOWN_AND_UP')
        device.press('KEYCODE_DPAD_DOWN','DOWN_AND_UP')
        device.press('KEYCODE_DPAD_DOWN','DOWN_AND_UP')
        device.press('KEYCODE_DPAD_RIGHT','DOWN_AND_UP')
        device.press('KEYCODE_DPAD_RIGHT','DOWN_AND_UP')
        device.press('KEYCODE_DPAD_DOWN','DOWN_AND_UP')
        device.press('KEYCODE_P','DOWN_AND_UP')
        device.press('KEYCODE_DPAD_DOWN','DOWN_AND_UP')
        device.press('KEYCODE_DPAD_CENTER','DOWN_AND_UP')
        device.press('KEYCODE_1','DOWN_AND_UP')
        device.press('KEYCODE_2','DOWN_AND_UP')
        device.press('KEYCODE_3','DOWN_AND_UP')
        device.press('KEYCODE_1','DOWN_AND_UP')
        device.press('KEYCODE_1','DOWN_AND_UP')
        device.press('KEYCODE_CALL','DOWN_AND_UP')
        device.press('KEYCODE_ENDCALL','DOWN_AND_UP')
        device.press('KEYCODE_HOME','DOWN_AND_UP')
```

Ker je izvajanje v simulatorju počasno se vse akcije odvijajo počasi in smo lahko natančno opazovali dogajanje. Občasno se je zgodilo, da si je Android zapomnil določeno stanje kot posledico zakasnjene akcije in zato v naslednjih ponovitvah obtičal na tem mestu. Napak ni bilo in test se je uspešno končal.

Na aparatu, ki hitreje opravi akcije, je bilo težje slediti dogajanju. Občasno so se pojavila kaka opozorila, ki pa se v nadaljevanju niso več prikazovala. Obnašanje telefona se je v prvi minuti opazno upočasnilo do neke točke in tam ostalo vse do konca testa. Tudi ta test se je zaključil brez težav.

V testu je bilo opazno, da je pošiljanje akcij operacija s potrditvijo, kar pomeni, da zahteve niso generirane neodvisno od prejemnika, ampak se naslednja zahteva pošlje šele ko je prejšnja zaključila obdelavo. Na ta način je zagotovljeno, da se vse poslane zahteve izvedejo, ne moremo pa doseči nasičenja zaradi prevelike količine zahtevkov na časovno enoto. Razlog je tudi, da so računalniki neprimerno hitrejši od človeka v pošiljanju akcij. V našem primeru je v približno desetih minutah telefon prejel 17.000 akcij, človek bi jih lahko generiral kvečjemu 6.000.

Stabilno in dodelano delovanje programske opreme kaže na zrelost projekta in resnost njegovih snovalcev in razvijalcev.

4. FTA (Andrejak, Cvenkel, Jež, Struna)

4.1. Analiza drevesa napak – FTA

Analiza drevesa napak (FTA, angl. fault tree analysis) je analiza odpovedi, kjer je neželeno stanje nekega sistema analizirano z uporabo Boolove logike, ki sestavlja serijo nižjih dogodkov. Ta analitična metoda se v glavnem uporablja za količinsko določanje verjetnosti pojava nevarnosti odpovedi. Večinoma so za izdelavo analize drevesa napak zadolženi inženirji s dobrim poznavanjem obravnavanega sistema.

Sprva je bil razvoj FTA namenjen projektom, pri katerih ni smelo prihajati do napak (jedrske elektrarne, letalstvo, ipd.). FTA je bil razvit s strani podjetja Bell Telephone Laboratories za potrebe ameriškega letalstva. Kasneje je FTA naprej razvijala družba Boeing Company, obsežnejše pa so ga uporabljale tudi ameriške jedrske elektrarne.

Izdelava FTA za velik sistem je lahko drag in časovno potraten postopek. Zato je pomembno, da si delo poenostavimo tako, da problem razbijemo na več manjših podproblemov, saj je analiza le teh lažja in tako zmanjšamo možnost napak.

Analiza drevesa napak se največkrat uporablja za:

- iskanje nevarnih, kritičnih komponent
- potrditve zahteve izdelka
- certificiranje zanesljivosti izdelka
- določanje varnosti izdelka
- preiskave nesreč in nepričakovanih dogodkov
- določanje načrtovanih sprememb
- identifikacijo vzrokov in posledic dogodkov
- iskanje običajnih napak

4.2. Diagram drevesa napak

V vsakem sistemu se je potrebno zavedati možnosti napak. Pri izdelavi drevesa napak je predpostavljeno, da je možnost delnega ali popolnega delovanja večja od možnosti delne ali popolne odpovedi, saj bi bila v nasprotnem primeru izdelava drevesa napak počasnejša od izdelave drevesa delovanja.

Diagram drevesa napak je logični bločni diagram, ki kaže stanje sistema (glavni dogodek) v odvisnosti od njegovih komponent (dogodkov). V korenu diagram nastopa nezaželeni glavni dogodek, listi v drevesu pa nastopajo kot posamezni dogodki, ki preko logičnih vrat pripeljejo do nezaželjenega glavnega dogodka. V listih nastopajo odpovedi opreme, človeški faktor in ostali faktorji, na katere nimamo vpliva(vremenski pogoji).

Sistem je bolj zanesljiv, če ima diagram drevesa napak čimveč AND vrat in čim manj OR vrat. Iz tega sledi da je cilj zanesljivega sistema da ima v FTA drevesu čimmanj OR vezav in čimveč večvhodnih AND vezav (redundanca).

Koncept gradnje diagrama drevesa napak je od zgoraj navzdol (angl. top-down). Kot zgled drevesa si oglejte slike v naslednjih poglavjih.

Najpogostejše napake pri gradnji drevesa napak so:

- uporaba prevelikega področja za glavni dogodek, kar pripelje do velikega in kompleksnega diagrama drevesa
- uporaba različnih nomenklatur za enake dogodke, kar onemogoča iskanje dogodkov, ki se ponavljajo v več vejah drevesa
- uporaba enake nomenklature za podobne, vendar različne dogodke, kar povzroči določitev istega dogodka za več scenarijev, vendar ti dogodki nastopijo ob različnih predpogojih
- razdelitev drevesa na veje po električnih, mehaničnih in strukturnih podsistemih, pozabi pa se na vmesnike in povezovanja v celoto.

Pri tem velja omeniti še slabosti FTA:

- eno FTA drevo analizira samo en način odpovedi (potreba po več FTA drevesih)
- vpliv subjektivnosti analitika
- zahteva veliko znanja, predvsem za statistične ocene

4.3. Metodologija

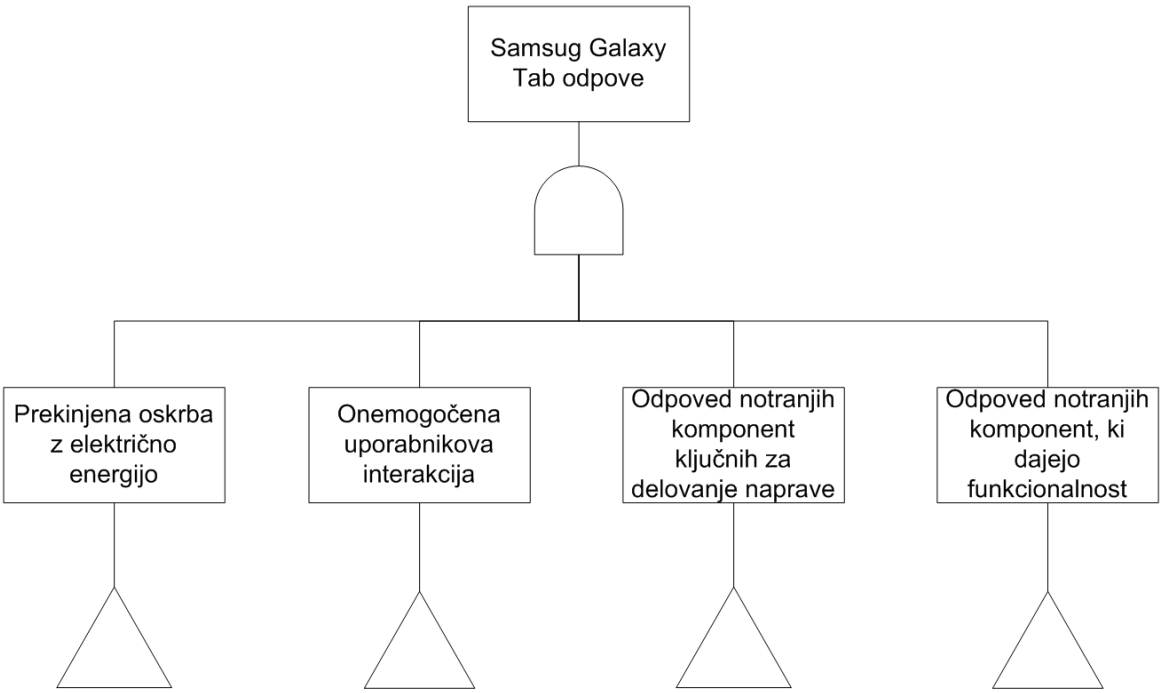
FTA je deduktivna analiza napak, ki temelji na enem glavnem neželenem dogodku in nam poda metodo za določanje vzroka za ta dogodek. Glavni neželeni dogodek je v korenu drevesa in večinoma pripelje do popolne, delne ali katastrofalne odpovedi sistema. Za glavni dogodek lahko vzamemo npr. strmoglavljenje potniškega letala, odtekanje hladilne tekočine pri hlajenju jedrskega reaktorja, neuspešen vžig avtomobila, odpoved računalniškega sistema v podjetju itd. V našem primeru prenehanje delovanja naprave Samsung Galaxy TAB.
Glavni dogodek je samo eden in vsi ostali dogodki pripeljejo do glavnega neželenega dogodka. Nato se vsak dogodek, ki bi lahko pripeljal do napake, doda drevesu napak kot zaporedje logičnih izrazov. Za vsak dogodek je potrebno določiti verjetnost njegovega pojava, ki pa jo je v praksi zaradi dragih testiranj zelo težko določiti. Ob vseh poznanih podatkih računalniški programi izračunajo verjetnost odpovedi podsistemov ali sistema.

4.4. Postopek

Koraki FTA so naslednji:

- določi objekt opazovanja
- določi n načinov odpovedi (n odpovedi--> n FTA dreves)
- določi n drevesnih struktur
- za vsako drevesno strukturo stori naslednje:
  - analiziraj vsa vejanja
  - določi "minimal cut set" - vse podmnožice listov, ki lahko povzročijo korenski dogodek - odpoved
  - vsakemu lističu določi verjetnost pojavitve

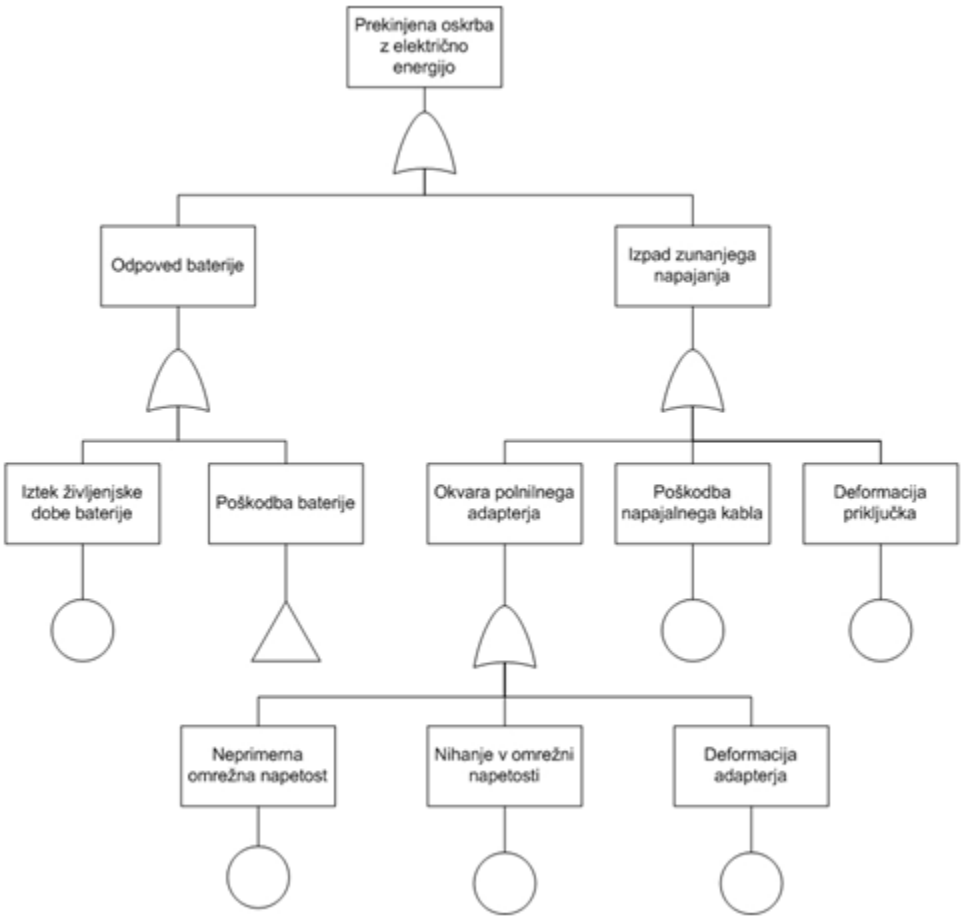
4.5. Vzroki in načini odpovedi



Za korenski dogodek FTA diagrama izberemo odpoved Samsung Galaxy TAB-a. Do odpovedi lahko pride zaradi različnih vzrokov. Dogodke lahko razdelimo na štiri glavne veje drevesa:

- Prekinjena oskrba z električno energijo
- Onemogočena uporabnikova interakcija
- Odpoved notranjih komponent ključnih za delovanje naprave
- Odpoved notranjih komponent, ki dajejo funkcionalnost.

4.6. Prekinjena oskrba z električno energijo

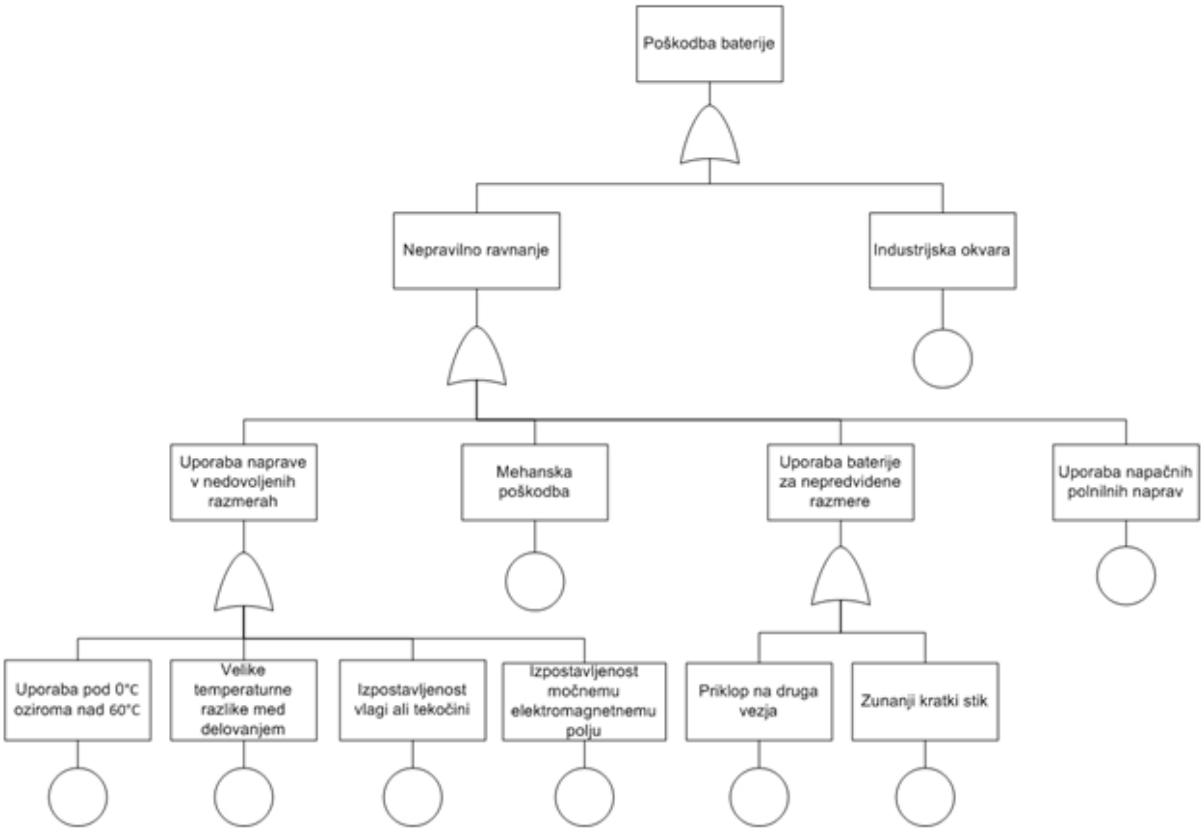


Samsung Galaxy TAB za napajanje uporablja 4000mAh baterijo, ki po proizvajalčevih navedbah omogoča do 7ur predvajanja videa (intenzivna uporaba) oziroma do 10 ur pogovorov. Naprava se napaja iz omrežnega napajalnika, ki je sestavljen iz dveh delov, to je adapterja in USB kabla. Pri tem lahko pride pri samem napajalniku do okvare na kablu ali napajalnem adapterju. Brez napajalnika je torej naprava dokaj neuporabna.
Glavna kritična dogodka sta ali odpoved baterije ali pa izpad zunanjega napajanja. V primeru drugega dogodka je naprava še uporabna do izpraznitve baterije, ob okvari baterije pa je nemudoma neuporabna.

4.6.1. Izpad zunanjega napajanja

Do odpovedi polnilnega adapterja lahko pride zaradi zgoraj naštetih vzrokov. Najverjetneje bo do tega prišlo zaradi poškodbe napajalnega kabla ali priključka, kar bi povzročilo deformacijo le teh. Možna je tudi okvara samega adapterja, do katere pa lahko pride zaradi neprimerne omrežne napetosti, saj se lahko ta v različnih državah razlikuje, priklop nanjo pa bi pomenil uničenje adapterja. Če bi v omrežju prišlo do velikega nihanja napetosti bi to lahko tudi pomenilo odpoved zunanjega napajanja. Zopet pa je tu možna tudi poškodba adapterja, zaradi česar uporaba ne bi bila več mogoča.

4.6.2. Odpoved baterije

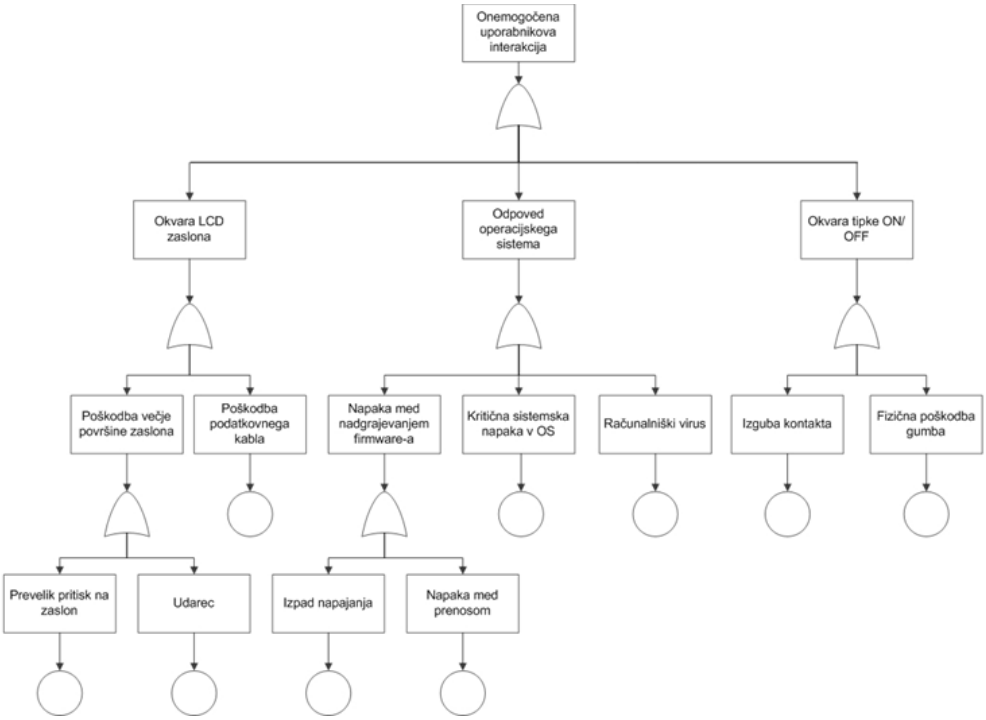


Do odpovedi baterije lahko pride zaradi izteka življenjske dobe baterije, kar pomeni da se je skozi čas njena kapaciteta tako zmanjšala, da ni več sposobna napajati telefona. Vsaka baterija ima namreč omejeno število polnjenj. Sčasoma pa tudi sama zmogljivost baterije pade. Druga možnost pa je ta, da se je baterija poškodovala in ni več sposobna opravljati svoje funkcije.

Do odpovedi baterije lahko pride zaradi nepravilnega ravnanja uporabnika oz. neupoštevanja navodil proizvajalca. Obstaja tudi verjetnost, da je prišlo do napake pri izdelavi. Ob kritični proizvodni napaki lahko baterija pod določenimi pogoji celo eksplodira.

Okvaro baterije lahko povzroči tudi malomarno ravnanje uporabnika. Možne so razne mehanske poškodbe zaradi grobega ravnanja, ki privedejo do odpovedi. Uporaba improviziranih polnilnih naprava ali napačnih polnilcev lahko tudi močno skrajša življenjsko dobo baterije, če jo ne celo uniči. Koriščenje baterije za namene, ki niso bili predvideni tj. napajanje drugih naprav (vezij) oziroma povzročitev zunanjega kratkega stika lahko ravno tako ogrozi brezhibno delovanje baterije in s tem celotne naprave. Potrebno je upoštevati tudi proizvajalčeva navodila glede okolja v katerem baterija deluje. Odpoved lahko povzročijo ekstremne zunanje temperature (nad 60°C, pod 0°C), nevarne so tudi hitre spremembe temperature okolja. Močna elektromagnetna polja lahko uničijo tako baterijo, kot seveda vse ostale elektronske komponente naprave. Velika vlažnost oziroma padec naprave v vodo je ravno tako lahko usoden tako za baterijo kot celotno napravo.

4.7. Onemogočena uporabnikova interakcija



Za normalno (nemoteno) uporabo je pomembno, da ne pride do poškodb ključnih delov naprave. Ob poškodbi oz. okvari teh komponent, je uporaba naprave zelo otežena ali celo nemogoča. Ker zaslon prevzema veliko večino vhodno-izhodnih funkcij lahko rečemo, da je ena najpomembnejših komponent naprave, zato mora vedno delovati brezhibno. Ob okvari LCD zaslona je naprava praktično neuporabna.

4.7.1. Okvara LCD zaslona

Do okvare zaslona lahko pride zaradi poškodbe ali slabega stika s podatkovnim kablom, bolj verjetno pa zaradi uporabnikove neprevidnosti. Do poškodb površine zaslona lahko pride zaradi premočnega pritiska ali močnega udarca. Poleg poškodbe površine zaslona lahko pride tudi do okvare osvetlitve, kar zelo oteži uporabo naprave v temi oz. pri močnem soncu. Vzrok za okvaro osvetlitve je lahko udarec ali izpostavljenost vlagi.

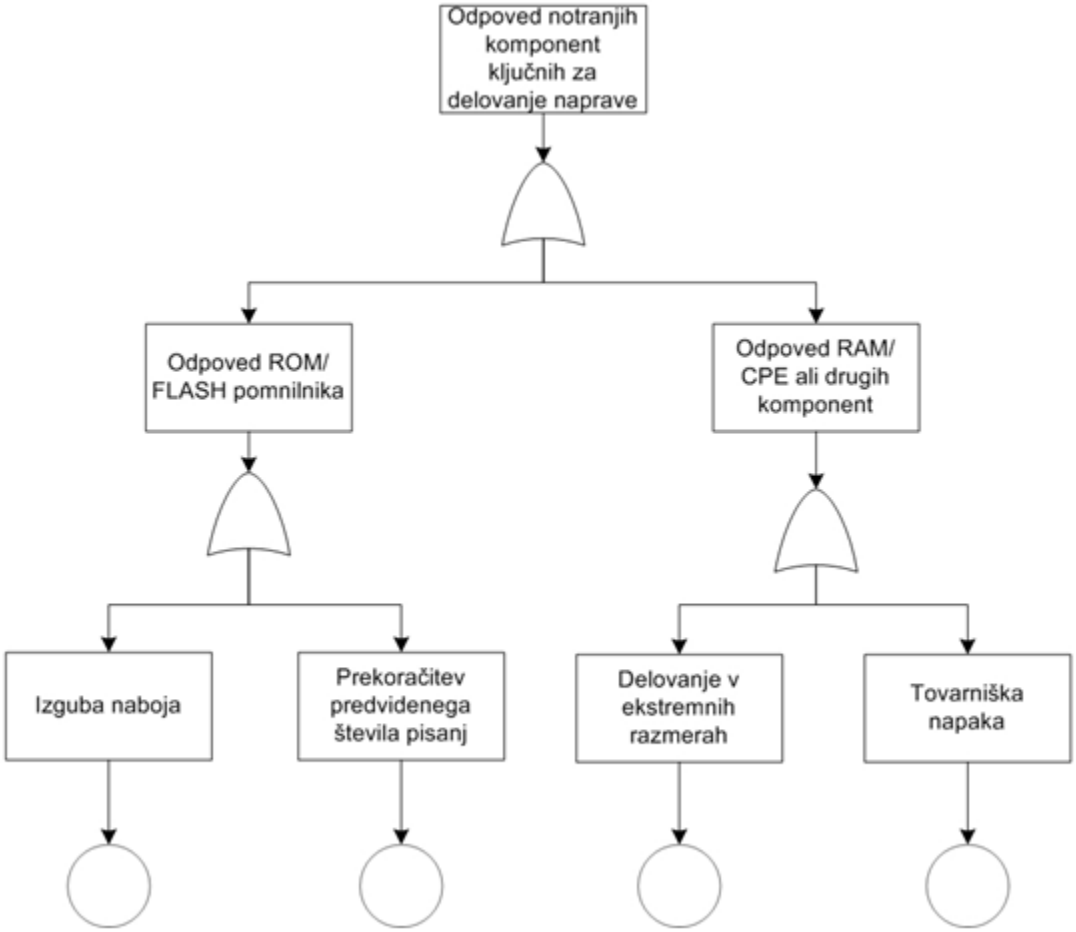
4.7.2. Odpoved operacijskega sistema

Do prenehanja delovanja operacijskega sistema lahko pride zaradi neuspešne (prekinjene) nadgradnje firmware-a, do katere pride zaradi izpada napajanja ali napake pri prenosu podatkov med posodobitvijo. Odpoved operacijskega sistema lahko povzroči tudi okužba z virusom ali pa nepreizkušena napačno delujoča programska koda, ki je lahko vzrok za kakšno kritično sistemsko napako.

4.7.3. Okvara tipke za vklop

Interakcija z napravo bi bila otežena ali celo onemogočena če pride do okvare tipke za vklop naprave. Do okvare lahko pride zaradi izgube kontakta (dotrajanost, slaba prevodnost) ali fizične poškodbe (zlom).

4.8. Odpoved notranjih komponent ključnih za delovanje naprave

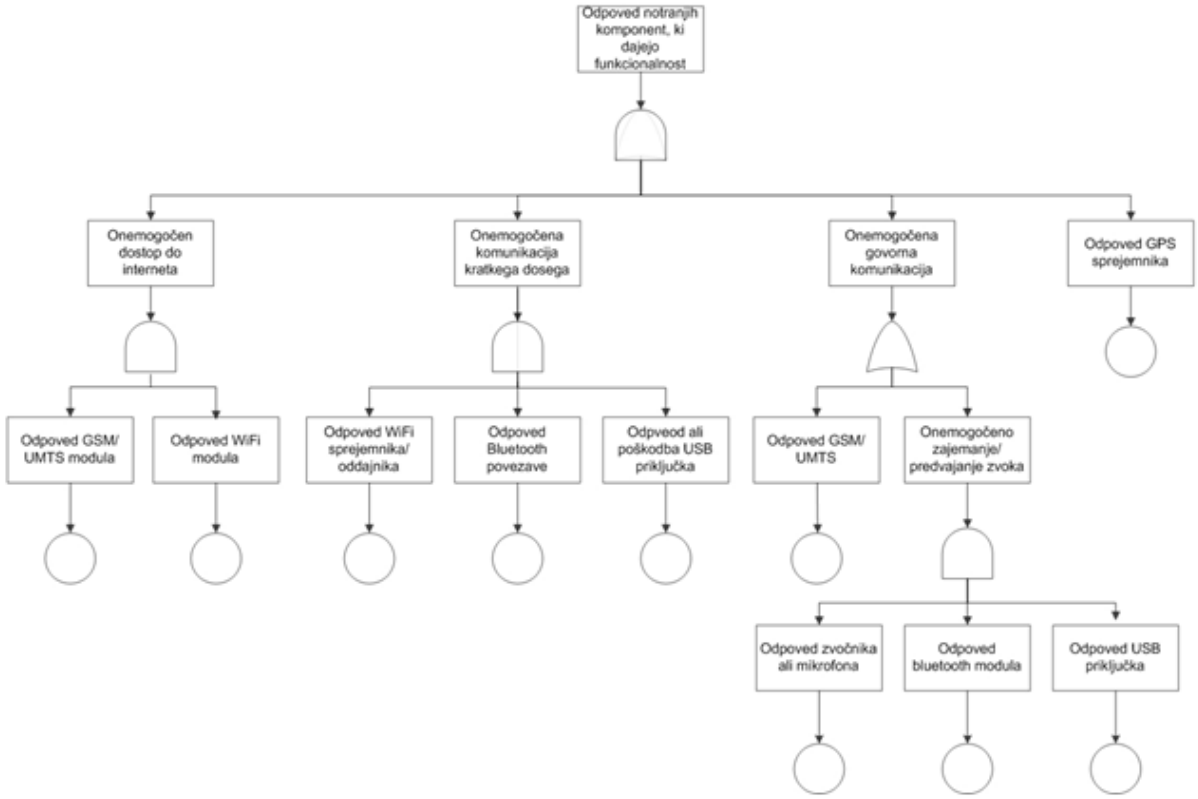


Kot vsak drug računalniški sistem tudi Samsung Galaxy TAB sestavljajo CPE in pomnilniki, ki so ključnega pomena za delovanje naprave. Okvara oz. odpoved delovanja teh komponent pomeni prenehanje delovanja naprave.

Naprava ima vgrajen procesor ARM 1 GHz Cortex A8 z 1GHz taktom ure. Kot integrirano vezje je zanesljiva komponenta, do okvare pa lahko pride zaradi tovarniške napake ali pa delovanja v ekstremnih razmerah.

Pomnilnik nam lahko odpove zaradi izgube naboja, starosti, prekomernega števila brisanj oz. pisanj, ekstremnih razmer ali pa celo tovarniške napake. V napravo je vgrajen RAM pomnilnik velikosti 512 MB. Za hranjenje podatkov je vgrajen tudi Flash pomnilnik velikosti 16GB, ki pa za delovanje sistema ni ključnega pomena.

4.9. Odpoved notranjih komponent, ki dajejo funkcionalnost napravi



Pri Samsung Galaxy TAB-u je potrebno upoštevati tudi odpovedi pri katerih naprava izgubi katero od funkcionalnosti. Določenim uporabnikom lahko postane naprava ob izgubi ene od njih neuporabna. Poleg velikega števila funkcij ki jih Samsung Galaxy TAB ponuja smo v grafu upoštevali tiste ki jih v splošnem izkorišča največ uporabnikov, vedeti pa moramo da za določenega uporabnika naprava lahko postane neuporabna tudi ob odpovedi katere od ostalih funkcionalnosti, ki niso upoštevane. Funkcionalnosti ki smo jih upoštevali so:

- onemogočen dostop do interneta
- onemogočena komunikacija kratkega dosega
- onemogočena govorna komunikacija
- odpoved GPS sprejemnika

4.9.1. Onemogočen dostop do interneta

Do odpovedi dostopa do interneta lahko pride zaradi okvare GSM/UMTS sprejemnika oz. oddajnika ali krmilnika. Pri odpovedi WI-Fi modula pa izgubimo dostop do interneta preko dostopnih točk. Vzroki za okvare teh naprav so najpogosteje slaba kakovost naprav ali fizične poškodbe.

4.9.2. Onemogočena komunikacija kratkega dosega

Komunikacija na kratke razdalje lahko poteka preko USB kabla, WI-FI ali Bluetooth povezave na neko drugo napravo, ki to omogoča. Ta funkcionalnost se izkorišča za prenos podatkov iz ene naprave na drugo. Vzroki za odpoved Bluetooth modula in USB komunikacije so podobni kot pri GSM/UMTS ter WI-Fi modulu.

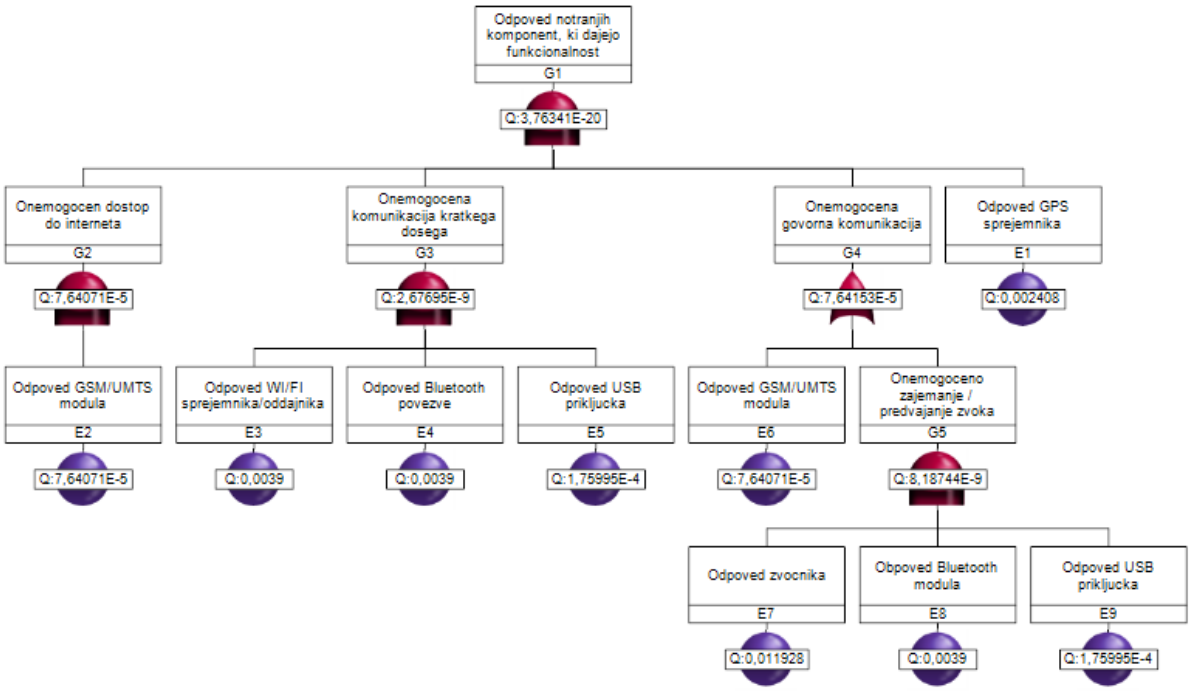
4.9.3. Onemogočena govorna komunikacija

Ob odpovedi GSM/UMTS modula je onemogočena tudi govorna komunikacija. Ta je odvisna tudi od delovanja zvočnika in mikrofona. V kolikor katera od teh dveh naprav odpove lahko govorno komunikacijo opravimo le preko slušalk in mikrofona, ki ju povežemo lahko preko Bluetooth povezave ali ¼ inch vhoda. Ob odpovedi vseh omenjenih opcij je govorna komunikacija onemogočena.

4.9.4. Odpoved GPS sprejemnika

Zadnja od omenjenih funkcionalnosti je GPS sprejemnik. Ta omogoča satelitsko določanje položaja naprave. Če se nam GPS sprejemnik pokvari je ta funkcija onemogočena. Najpogostejši razlogi za okvaro so fizične poškodbe, slaba kvaliteta ali ekstremi pogoji za delovanje.

4.9.5. Izračun verjetnosti odpovedi



4.10. Zaključek FTA

Postopek FTA analize vključuje še verjetnosti pojavitve odpovedi na vsakem od listov, ter verjetnost odpovedi celotnega sistema, ki jih nismo dodali. Izračunali smo jih le za drevo, pri katerem smo imeli podatke za verjetnosti odpovedi vseh komponent. Problem je da vpogleda v ostale podatke o verjetnosti odpovedi posameznih komponent naprave nimamo. Ti podatki so po navadi znani proizvajalcu.

Vseeno je pa izdelava diagramov odpovedi koristna, saj nam ti prikazujejo katere komponente in funkcionalnosti so za delovanje naprave pomembnejše in pri katerih komponentah lahko pričakujemo glavne vzroke odpovedi naprave. Kakšna je odvisnost med komponentami pa lahko razberemo iz hierarhije dogodkov. S pomočjo diagramov uporabnik lahko že vnaprej ve kako se bo naprava obnašala in na kaj mora biti posebno pozoren pri uporabi. V diagramih so dogodki najpogosteje povezani z vrati OR. Iz tega je razvidno, da skoraj vsak od dogodkov pripelje do odpovedi naprave. Le pri diagramu odpovedi komponent, ki dajejo funkcionalnost napravi smo uporabili AND vrata, saj okvara ene od komponent še ne pomeni odpoved cele naprave, lahko pa s tem močno zmanjša uporabnost. Je pa tukaj spet vprašanje namena uporabe, saj lahko odpoved neke funkcionalnosti nekomu ne povzroči nobenih težav, nekomu drugemu pa naredi napravo popolnoma neuporabno. Na to je potrebno paziti in upoštevati pri analizi modernih napravah s toliko različnimi nameni uporabe.

Vidimo torej da ima naprava zelo nizko redundanco. To je značilno za naprave ki so namenjene masovni prodaji, pri katerih ni pričakovana visoka zanesljivost.

5. FMEA (Berdajs, Bizjak, Krečič)

Izraz FMEA pomeni failure modes and effects analysis, kar v dobesednem prevodu pomeni analiza načinov in učinkov odpovedi. Gre za postopek, pri katerem identificiramo vse potencialne načine odpovedi, nato pa jih lahko zaradi njihove resnosti popolnoma izničimo ali pa le omilimo njihove posledice. Tak postopek je uporaben pri analizi dejanskih izdelkov, funkcionalnosti ali pa celo procesov. FMEA omogoča sistematično preučevanje potencialnih odpovedi, kar je dobro, saj to pri razvoju izdelka lahko prihrani mnogo stroškov. [5.1]

Metoda FMEA je bila prvič opisana v MIL-P-1629 standardu ameriške vojske leta 1949, kasneje pa je bila uporabljena tudi pri razvoju vesoljskega programa Apollo. Postopoma se je razširila tudi v prehrambno industrijo, avtomobilsko industrijo, proizvodnjo polprevodnikov, na področje programske opreme ter še na mnoga druga.

Pri načrtovanju izdelka so zelo pomembne lastnosti cena, varnost za uporabnika, zmogljivost, zanesljivost, vzdržljivost itd. Če poznamo slabosti nekega dizajna, jih lažje odpravimo in se osredotočimo na pomembnejše dele sistema, pri tem pa denarna sredstva bolje izkoristimo.

Veliko proizvajalcev FMEA analizo izvajajo skozi celoten življenjski cikel - ne le v času razvoja ampak tudi v času izrabe izdelka. Pridobljene izkušnje lahko nato uporabijo pri razvoju ostalih izdelkov.

5.1. Osnovni izrazi

Pri uporabi te metode se pogosto pojavljajo naslednji izrazi:
Odpoved – izguba določene funkcionalnosti pod nekimi pogoji
Način odpovedi – način, na katerega se zgodi odpoved
Učinek odpovedi – posledice odpovedi na funkcionalnost, funkcijo, stanje ali delovanje predmeta
Vzrok odpovedi – napaka v načrtovanju, procesu ali kvaliteti, ki vodi do odpovedi
Resnost odpovedi – najhujša posledica določene odpovedi
Lokalni učinek – učinek odpovedi na komponento na višjem nivoju
Končni učinek – učinek odpovedi na celoten sistem

5.2. Opis metode

Osnovni postopek FMEA metode bi lahko strnili v nekaj osnovnih korakov:

- Identifikacija vseh možnih načinov odpovedi.
- Ocena tveganja posameznih načinov odpovedi
- Rangiranje najbolj kritičnih odpovedi
- Ukrepanje z namenom zmanjšati učinek najbolj kritičnih možnih odpovedi ali jih popolnoma odpraviti
- Ponovna ocena tveganja, da vidimo kakšne učinke so imeli ukrepi

V 2. točki – ocenjevanje tveganja - imamo na izbiro več možnosti. Poslužujemo se lahko zelo pogosto uporabljene metode, ki se imenuje **Risk Priority Numbers** ali pa tudi ostalih, kot je npr. metoda **Criticality Analysis**.

5.2.1. Metoda Risk Priority Numbers

Risk Priority Numbers oz. **RPN** metoda temelji na tem, da najprej za vsak tip odpovedi, ki smo ga identificirali določimo tri parametre, vsakemu pa pripišemo vrednost od 0 - 10:

Occurrence (možnost pojavitve) – vrednost 1 – 10 dodelimo glede na to, kako pogosto lahko pride do nekega tipa odpovedi, kjer najnižja pomeni, da se ta tip odpovedi pojavi zelo redko, višja pa da se pogosteje. Na možnost pojavitve odpovedi vplivajo predvsem napake v dizajnu sistema.

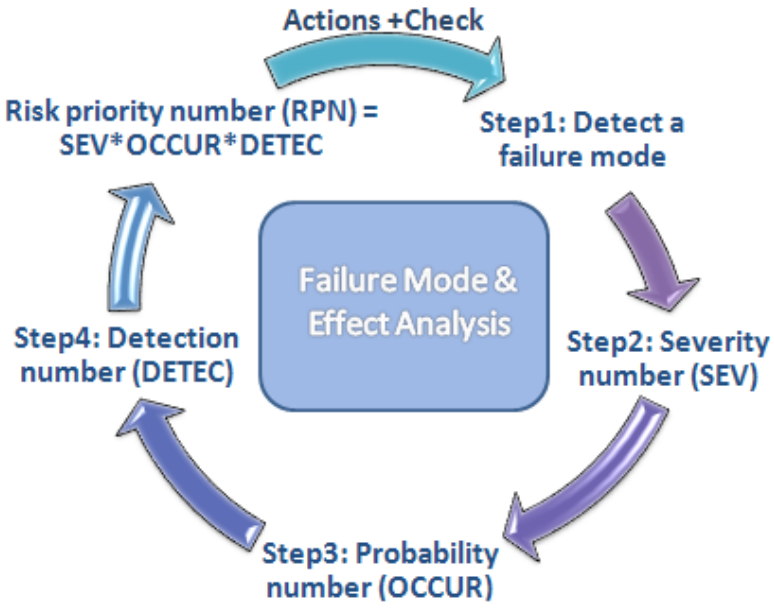
Severity (resnost) – pove kako resne posledice ima lahko pojavitev načina odpovedi. Nizka vrednost lahko pomeni, da posledic ni ali pa so zelo majhne, višja pa da lahko sistem delno ali v celoti preneha delovati ali pa celo postane nevaren za uporabnika.

Detection (možnost zaznavanja) – še zadnji parameter, ki ga moramo določiti je možnost zaznavanja pojavitve načina odpovedi. Nizka vrednost tu pomeni, da bomo odpoved zlahka zaznali, višja pa da težje oz. skoraj nemogoče.

Z vsemi tremi določenimi vrednostmi O, S in D sedaj lahko ocenimo kritičnost načinov odpovedi. Za vsak način odpovedi izračunamo vrednost RPN na način: glede na izračunane RPN, ki jih dobimo na način:

RPN = Severity * Occurrence * Detection

S tako izračunanimi RPN vrednostmi za vsak način odpovedi, lahko te rangiramo glede na RPN. Tisti načini odpovedi, ki imajo najvišji RPN so dobri kandidati za izboljšanja, ni pa seveda nujno, da je ravno odpoved z najvišjim RPN tudi najbolj kritična – potrebna je torej pazljivost.
Potrebno je omeniti še to, da kadarkoli izvedemo spremembe v sistemu, je potrebno RPN vrednosti ponovno izračunati. Tako lahko primerjamo učinke naših popravkov na načine odpovedi.



Slika 5.1: Shematski prikaz izvedbe FMEA z metodo RPN
http://en.wikipedia.org/wiki/File:FMEA.png

5.2.2. Metoda Criticality Analysis

FMEA metodo z analizo kritičnosti pogosto označimo kot **Failure Modes, Effects and Criticality Analysis oz. FMECA**. Je razširitev FMEA in se uporablja za prikaz verjetnosti načinov odpovedi v odvisnosti od resnosti posledic. S pomočjo rezultatov FMECA analize lahko identificiramo načine odpovedi z visokimi verjetnostmi odpovedi in visoko resnostjo posledic. FMECA se največ uporablja v vesoljskih programih in NATO vojaških aplikacijah, v drugih industrijah pa se večinoma uporabljajo razne variacije metode FMEA. [5.2]

Kriterij pri FMECA je *modal criticality number* C_m, ki ga izračunamo na podlagi naslednjih podatkov:

- λ_p - kako pogosto prihaja do načina odpovedi, ponavadi se uporablja predikcija po modelu MIL–HDBK–217, PRISM, ali drugih.
- α - delež načina odpovedi (med 0 in 1) med vsemi načini odpovedi za dotično komponento ali funkcijo.
- β - pogojna verjetnost, ki nam pove verjetnost, da bo način odpovedi pripeljal do ocenjene resnosti posledic (severity).
- t - čas trajanja faze misije.

Formula za izračun C_m je sledeča:

$$C_m = \lambda_p * \alpha * \beta * t$$

Po izračunu C_m se tabela uredi po resnosti posledic ter faktorju C_m. Iz tako urejene tabele lahko enostavno identificiramo kritične načine odpovedi, ki bi jih bilo treba odpraviti.

5.2.3. Predpriprave

Pri FMEA analizi je zelo pomembno, da izdelek, ki ga analiziramo dobro poznamo. Zavedati se moramo kateri načini uporabe so zaželeni in kateri niso. Potrebno je identificirati vse glavne komponente izdelka, ki zagotavljajo želene funkcionalnosti in tudi to, kako so povezane med sabo. Tako lahko vemo, kako odpoved ene komponente vpliva na delovanje drugih, ki so z njimi povezane. Samo strukturo komponent izdelka lahko predstavimo z bločnim diagramom ali pa z drevesno strukturo.
Na podlagi preteklih izkušenj in analize robustnosti moramo nato določiti tudi kriterije za vrednosti Occurence, Severity in Detection, na podlagi katerih bomo komponentam dodelili določen RPN.

5.3. Izvedba analize

Za izvedbo FMEA analize smo najprej želeli uporabiti programski paket Relex 2011 proizvajalca PTC. Na žalost se je izkazalo, da je preizkusna različica zelo omejena, tako da smo se odločili analizo izvesti ročno, torej s programom za urejanje preglednic – tako je vnos podatkov preprost, prav tako pa računanje RPN vrednosti in izdelava grafov.

V osnovi poznamo tri FMEA analize:

- **Procesna FMEA**
  Gre za analizo posledice odpovedi med procesom proizvodnje ali sestavljanja nekega izdelka na delovanje izdelka.
- **Funkcionalna FMEA**
  Gre za analizo želenih funkcionalnosti izdelka oz. storitve. Ugotavlja se, kaj bi se zgodilo, če bi določena funkcionalnost odpovedala in kakšne posledice bi imelo to na celoten izdelek oz. proces.

- **Komponentna FMEA**
  Tu se izvede analizo posledic odpovedi določenih komponent, ki zagotavljajo funkcionalnosti izdelku, tipično so to neke strojne komponente na nizkem nivoju.

Procesna analiza za nas ne pride v poštev, saj je izdelek že narejen, izvedli smo **komponentno** FMEA analizo. Pri komponentni analizi smo lahko preprosto identificirali glavne elektronske ali mehanske komponente izdelka, kar je bilo možno z odpiranjem ohišja in razstavljanjem ter iskanjem nadaljnjih informacij o komponentah preko spleta.

5.3.1. Kriteriji

Za vsak kriterij Occurence, Severity in Detection smo po lastni presoji določili predvidene verjetnosti pojavitve.

Occurence

Ocena	Kriterij	Ocenjena verjetnost odpovedi
1	odpoved skoraj neverjetna	Praktično nikoli
2	odpoved zelo malo verjetna	Zelo malo odpovedi
3	odpoved malo verjetna	Malo odpovedi
4, 5, 6	odpoved srednje verjetna	Občasne odpovedi
7, 8	odpoved visoko verjetna	Ponavljajoče odpovedi
9, 10	odpoved zelo verjetna	Zelo pogoste odpovedi

Severity

Ocena	Resnost odpovedi	Vpliv odpovedi
1	Nevidno	Ni vplivov
2	Ni razlike v delovanju (za povprečnega uporabnika)	Minimalen
3	Majhna razlika v delovanju (povprečni uporabnik ne opazi)	Majhen
4, 5, 6	Opazna razlika v delovanju (za povprečnega uporabnika moteče)	Srednji
7, 8	Izguba glavne funkcionalnosti (uporabniki so nezadovoljni)	Visok
9	Umakne se celotno serijo (uporabniki so zelo nezadovoljni)	Zelo visok

Detection

Ocena	Detekcija
1	Vedno mogoče zaznati odpoved
2	Skoraj vedno mogoče zaznati odpoved
3	Zaznavanje odpovedi zelo verjetno
4, 5, 6	Zaznavanje odpovedi srednje verjetno
7, 8	Zaznavanje odpovedi malo verjetno
9	Skoraj nikoli ni možno zaznati odpovedi
10	Ni možno zaznati odpoved

5.3.2. Komponentna analiza

Najprej smo identificirali vse ključne strojne komponente tabličnega računalnika in jih zapisali v obliki drevesne strukture. Nekatere komponente so vidne že na zunaj, večina pa jih je skritih v notranjosti. Potrebne podatke smo dobili s pregledom samega izdelka ter iz različnih virov na internetu, kjer je Galaxy TAB razstavljen in so identificirane tudi njegove glavne komponente v notranjosti.
Spodaj so predstavljene glavne komponente Samsung Galaxy TAB-a, ki so urejene v drevesno strukturo:

1. Samsung Galaxy TAB

Ohišje
Akumulator 4000 mAh
Mikrofon
Stereo zvočnika
Kamera 3.2 MP
Kamera 1.3 MP
Gumbi
Konektorji

- 3.5 mm jack
- SIM kartica
- microSD kartica
- Priključna postaja (USB)

Zaslon

- LCD ekran
- Senzor dotikov

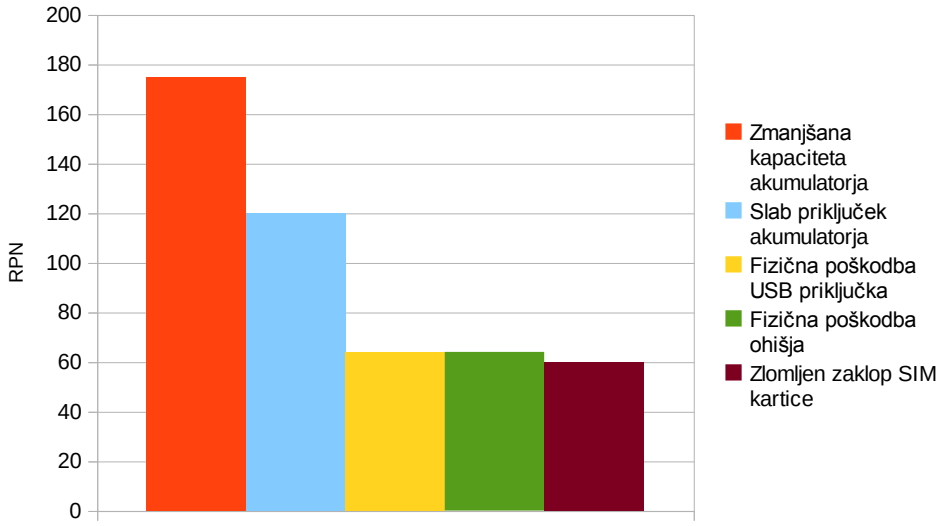
Pomnilniki

- 512 MB RAM
- 16 GB Flash SanDisk

Matična plošča

- Procesor ARM Cortex Hummingbird 1GHz + 3 nivoji pomnilnika
- Vezje za nadzor napajanja Maxim 8998
- GPS sprejemnik Broadcom BCM4751
- Modem HSDPA/HSUPA/EDGE Infeon
- Bluetooth/FM/WLAN modul Broadcom BCM4329
- Žiroskop STMicroelectronics L3G4200D
- Krmilnik zaslona na dotik ATMEL MXT224

Po izdelavi tabele, ki je priložena na koncu, kjer smo identificirali vse možne načine odpovedi in določili vrednosti kriterijev O, S in D smo izračunali vrednosti RPN. Vse načine odpovedi smo rangirali glede na izračunano RPN in dobili 5 najvišjih vrednosti, ki so predstavljene na grafu.



Slika 5.2: Najvišje vrednosti RPN

Iz grafa lahko razberemo pri katerih komponentah so odpovedi najbolj kritične. To so zmanjšana kapaciteta akumulatorja, slab priključek akumulatorja, fizična poškodba USB priključka, fizična poškodba ohišja in zlomljen zaklop SIM kartice.

Do zmanjšane kapacitete akumulatorja privede več razlogov: največkrat je to posledica nepravilnega polnjenja akumulatorja, ko akumulatorja ne napolnimo do konca oz. ga ne porabimo, ampak ga večkrat polnimo po malo - vsak akumulator ima neko omejeno število ciklov polnjenja.

Slab priključek za polnjenje akumulatorja postane največkrat zaradi grobega ravnanja uporabnika, možna pa je tudi krivda proizvajalca, ki uporablja cenene priključke. To odpoved zelo hitro opazimo, saj imamo težave pri polnjenju akumulatorja.

Za poškodbo USB priključka podobno kot pri priključku za polnjenje akumulatorja je odgovoren uporabnik sam, to se največkrat zgodi pri grobem ravnanju z napravo.

Ohišje oklepa večino komponent, ki so shranjene noter, zato pri raznih padcih tudi utrpi največje poškodbe. Ohišje mora biti kakovostno, da material ob padcu ne poči in tako varuje komponente. V večini primerov pri padcih se največkrat sname pokrov akumulatorja in akumulator pade iz ležišča.

Zlomljen zaklop SIM kartice pomeni da imamo probleme pri vstavljanju SIM kartice. To odpoved lahko pripišemo grobemu, nespretnemu ravnanju uporabnika, zgodi se pri pogostem menjavanju SIM kartice.

5.4. Zaključek FMEA

Rezultati FMEA komponentne analize so pokazali, da je za večino odpovedi odgovoren uporabnik sam, naj si bo to nepravilno polnjenje akumulatorja ali pa poškodbe raznih priključkov. Analizo smo opravili na podlagi naših, subjektivnih ocen ter izkušenj z napravami s podobnimi komponentami. Bolj točne podatke bi dobili od serviserjev, mogoče če bi povprašali na kakšnem forumu na internetu. Število RPN bi zmanjšali s kvalitetnejšimi, robustnimi materiali, tako bi dobili boljšo oceno RPN.

Izdelek/funkcija	Odpoved	Učinek odpovedi	S (resnost)	Razlogi	O (pojavitve)	D (zaznavanje)	RPN	Akcije
1 – Samsung Galaxy Tab								
1.1 – Ohišje								
1.1.1 – Zaščita notranjih komponent								
	Fizična poškodba	Nedelovanje naprave	8	Udarec naprave	4	2	64	Odpornejše ohišje
1.2 – Akumulator 4000 mAh								
	Zmanjšana kapaciteta	Krajše delovanje brez polnjenja	5	Dotrajanost, vročina	7	5	175	Kvalitetnejši akumulator
	Slab priključek akumulatorja	Težave pri polnjenju	8	Grobo ravnanje, slaba kvaliteta	5	3	120	Kvalitetnejši priključek
	Vžig akumulatorja	Uničenje naprave	10	Vročina	1	1	10	Toplotna izolacija
1.3 – Mikrofon								
	Prekinjen priključek	Nezaznavanje zvoka	8	Nekvalitetni spoji, fizična poškodba	2	1	16	Kvalitetnejši priključek
1.4 – Stereo zvočnika								
	Prekinjen priključek	Predvajanje zvoka ne deluje	9	Nekvalitetni spoji, fizična poškodba	2	1	18	Kvalitetnejši priključek
1.5 - Kamera 3.2 MP (na zadnji strani ohišja)								
	Poškodba leče	Zamegljena slika	5	Fizična poškodba	3	2	30	Odpornejša leča
	Odpoved CCD senzorja	Nedelovanje kamere	8	Napaka na vezju, slaba kvaliteta	1	1	8	Kvalitetnejša integracija
1.6 – Kamera 1.3 MP (na sprednji strani ohišja)								
	Poškodba leče	Zamegljena slika	5	Fizična poškodba	3	2	30	Odpornejša leča
	Odpoved CCD senzorja	Nedelovanje kamere	7	Napaka na vezju, slaba kvaliteta	1	1	7	Kvalitetnejša integracija
1.7 – Gumbi								
1.7.1 – Vklop/izklop								
	Gumb ne spreminja stanja	Moten nadzor nad napravo	10	Fizična poškodba	4	1	40	Kvalitetnejši gumbi
1.7.2 – Glasnost								
	Gumb ne spreminja stanja	Oteženo spreminjanje glasnosti	7	Fizična poškodba	4	1	28	Kvalitetnejši gumbi
1.7.3 – 4 gumbi na dotik								
	Gumb ne spreminja stanja	Moten nadzor nad napravo	9	Fizična poškodba	4	1	36	Kvalitetnejši gumbi
1.8 - Konektorji								
1.8.1 – 3.5 mm jack								
Vhod za slušalke	Poškodovan priključek	Izhod za zvok ne deluje	6	Fizična poškodba	5	1	30	Kvalitetnejši priključek
1.8.2 – SIM kartica								
	Zlomljen zaklop	SIM kartica ne ostane na mestu	10	Grobo ravnanje, slaba kvaliteta	3	2	60	Kvalitetnejši material
	Ni stika na kontaktih	Nezaznavanje SIM kartice	9	Nekvalitetni spoji, fizična poškodba	4	1	36	Kvalitetnejša integracija
1.8.3 – microSD kartica								
Razširitvena kartica	Zlomljen zaklop	Kartica ne ostane na mestu	8	Grobo ravnanje, slaba kvaliteta	3	2	48	Kvalitetnejši akumulator
	Ni stika na kontaktih	Nezaznavanje SD kartice	9	Nekvalitetni spoji, fizična poškodba	4	1	36	Kvalitetnejša integracija
1.8.4 – Priključna postaja (USB)								
Polnjenje akumulatorja in povezava USB	Fizična poškodba	Moteno polnjenje in povezava	8	Grobo ravnanje, slaba kvaliteta	4	2	64	Kvalitetnejši priključek
	Ni stika na kontaktih	Nedelovanje polnjenja in poveza	10	Nekvalitetni spoji, fizična poškodba	2	1	20	Kvalitetnejša integracija
1.9 – Zaslon								
1.9.1 – LCD ekran								
	Odpoved delov zaslona	Nepopolna slika	8	Napaka na vezju, fizična poškodba	1	2	16	Kvalitetnejša integracija
	Odpoved celega zaslona	Ni prikaza slike	10	Napaka na vezju, fizična poškodba	1	1	10	Kvalitetnejša integracija
	Nedelujoča osvetljava	Temen zaslon	9	Napaka na vezju	1	2	18	Kvalitetnejša integracija
1.9.2 – Senzor dotikov								
	Nenatančno zaznavanje	Moten nadzor naprave	8	Ukrivljenost zaslona	1	2	16	Kvalitetnejši material
	Neodzivanje na dotike	Nemogoč nadzor naprave	10	Napaka na vezju	2	1	20	Kvalitetnejša integracija
1.10 – Pomnilniki								
1.10.1 – 512 MB RAM								
Delovni pomnilnik	Pomnilnik ne deluje	Naprava ne deluje	10	Napaka na vezju, slaba kvaliteta	1	1	10	Kvalitetnejša integracija
1.10.2 – 16 GB Flash SanDisk								
Trajni pomnilnik	Pomnilnik ne deluje	Naprava ne deluje	10	Napaka na vezju, slaba kvaliteta	1	1	10	Kvalitetnejša integracija
1.11 – Matična plošča								
1.11.1 – Procesor ARM Cortex Hummingbird 1GHz + 3 nivoji pomnilnika								
	Procesor ne deluje	Naprava ne deluje	10	Napaka na vezju	1	1	10	Kvalitetnejša integracija
1.11.2 – Vezje za nadzor napajanja Maxim 8998								
	Napačna napetost	Naprava ne deluje	10	Napaka na vezju	1	1	10	Kvalitetnejša integracija
	Ne daje napetosti	Naprava ne deluje	10	Napaka na vezju	1	1	10	Kvalitetnejša integracija
1.11.3 – GPS sprejemnik Broadcom BCM4751								
	Daje nepravilne podatke	Napačno delovanje GPS	6	Okvara antene, slabi spoji	2	3	36	Kvalitetnejša integracija
	Odpoved modula	Nedelovanje GPS funkcij	8	Okvara sprejemnika ali antene	1	2	16	Kvalitetnejša integracija
1.11.4 – Modem HSDPA/HSUPA/EDGE Infeon								
	Odpoved modula	Prenos podatkov ne deluje	8	Okvara antene, slabi spoji	1	2	16	Kvalitetnejša integracija
1.11.7 – Bluetooth/FM/WLAN modul Broadcom BCM4329								
	Odpoved modula	Prenos podatkov ne deluje	7	Okvara antene, slabi spoji	2	2	28	Kvalitetnejša integracija
1.11.8 – Žiroskop STMicroelectronics L3G4200D								
	Odpoved modula	Nezaznavanje orientacije	6	Fizična poškodba, slabi spoji	2	1	12	Kvalitetnejša integracija
1.11.9 – Krmilnik zaslona na dotik ATMEL MXT224								
	Odpoved modula	Nemogoč nadzor naprave	10	Napaka na vezju	1	1	10	Kvalitetnejša integracija

6. Markovska analiza (Adzievski, Paspalovski, Uršič)

6.1. Definicija markovske analize

Markovska analiza je statistična metoda, ki se uporablja za analizo zanesljivosti delovanja sistemov, ki so sestavljeni iz medsebojno povezanih komponent. Najprej bom razložil nekaj osnov, ki so potrebne za razumevanje principa markovske analize. Prvi pojem, ki ga srečamo je markovski model. Tega srečamo pri verjetnosti teoriji in predstavlja stohastičen model, ki predpostavlja osnovno matematično lastnost, ki se glasi takole:

$$\Pr(X_{n+1} = x | X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = \Pr(X_{n+1} = x | X_n = x_n).$$

Povedano drugače, vsako trenutno stanje sistema je odvisno samo od prejšnjega stanja sistema. V praktičnem smislu je to zelo uporabna lastnost, saj reši problem pomnjenja dolge zgodovine stanj opazovanega sistema.

Drugi pomemben pojem so Markovske verige. To je najenostavnejši primer Markovskega modela. Omogoča modeliranje sistema za naključno spremenljivko, ki se spreminja skozi čas. Oziroma z upoštevanjem zgornje Markovske lastnosti je verjetnostna porazdelitev naključne spremenljivke odvisna samo od verjetnostne porazdelitve spremenljivke predhodnega stanja.

Drugi primer Markovskega modela je Markovski proces. Bistvena razlika med modeloma je, da so stanja in čas pri Markovskih verigah diskretna medtem ko so pri Markovskih procesih stanja sistema diskretna in čas zvezen.

Opozoriti je potrebno na stohastičnost sistemov s katerimi se ukvarjamo. Razume se, da je obnašanje naprave, ki jo analiziramo nepredvidljivo oz. da je pri vseh izračunih vključen pojem naključnosti. Torej je kriterij po katerem smo uporabljali model pogojen z nekimi verjetnostmi med prehodi stanj sistema. Te verjetnosti so bile izračune s pomočjo MIL standarda. V tej sekciji se s tem nismo ukvarjali, ampak smo imeli vse te podatke dane.

Kot se lahko vidi v tej seminarski nalogi, obstaja več različnih tehnik za ocenjevanje zanesljivosti delovanja opazovanega sistema. Vsaka tehnika pa ima nekatere prednosti in nekatere pomanjkljivosti pred katero drugo tehniko. Naj omenimo prednosti Markovske analize:

- Preprost modelirni pristop. Kljub vsej matematični kompleksnosti modela (eksplozija stanj, ki bo kasneje razložena) je realizacija v programskem okolju, ki ga uporabljamo relativno enostavna.
- Manipulacija oz. dodatno spreminjanje konfiguracije modela je precej enostavno. Tukaj je mišljena povezava med stanji sistema, uteži verjetnosti, dodajanje ali odstranjevanje stanj itd.
- Tukaj lahko kompleksnejše dogodke razdelimo na poddogodke, ki so medseboj odvisni. Tako lahko s pomočjo generiranja sekvence poddogodkov sestavimo kompleksen dogodek.

Kompleksnost modela je pomemben pojav pri Markovski analizi, kateremu je primerno posvetiti nekaj besed. Kompleksnost je velikost množice vseh možnih stanj sistema. Najlaže se zadeva ilustrira na primeru. Recimo, da imamo sistem, ki ima v nekem trenutku N stanj. Želimo mu dodati neko komponento, ki se lahko nahaja v M stanjih. Takoj, ko sistemu dodamo to komponento bo število vseh stanj sistema naraslo v N*M. Torej lahko vidimo, da kompleksnost sistema raste eksponentno s številom komponent sistema.

6.2. Analiza našega izdelka

6.2.1. Funkcionalnost naprave

Najprej moramo naš sistem razdeliti na komponente glede na pomembnost, ki jo prispevajo k delovanju celotnega sistema.

Nekatere komponente so z vidika delovanja naprave kritično. To pomeni, da ob odpovedi takšne komponente sistem nemudoma preide v stanje nedelovanja oz. v angleščini "Fail". Te komponente so sledeče:

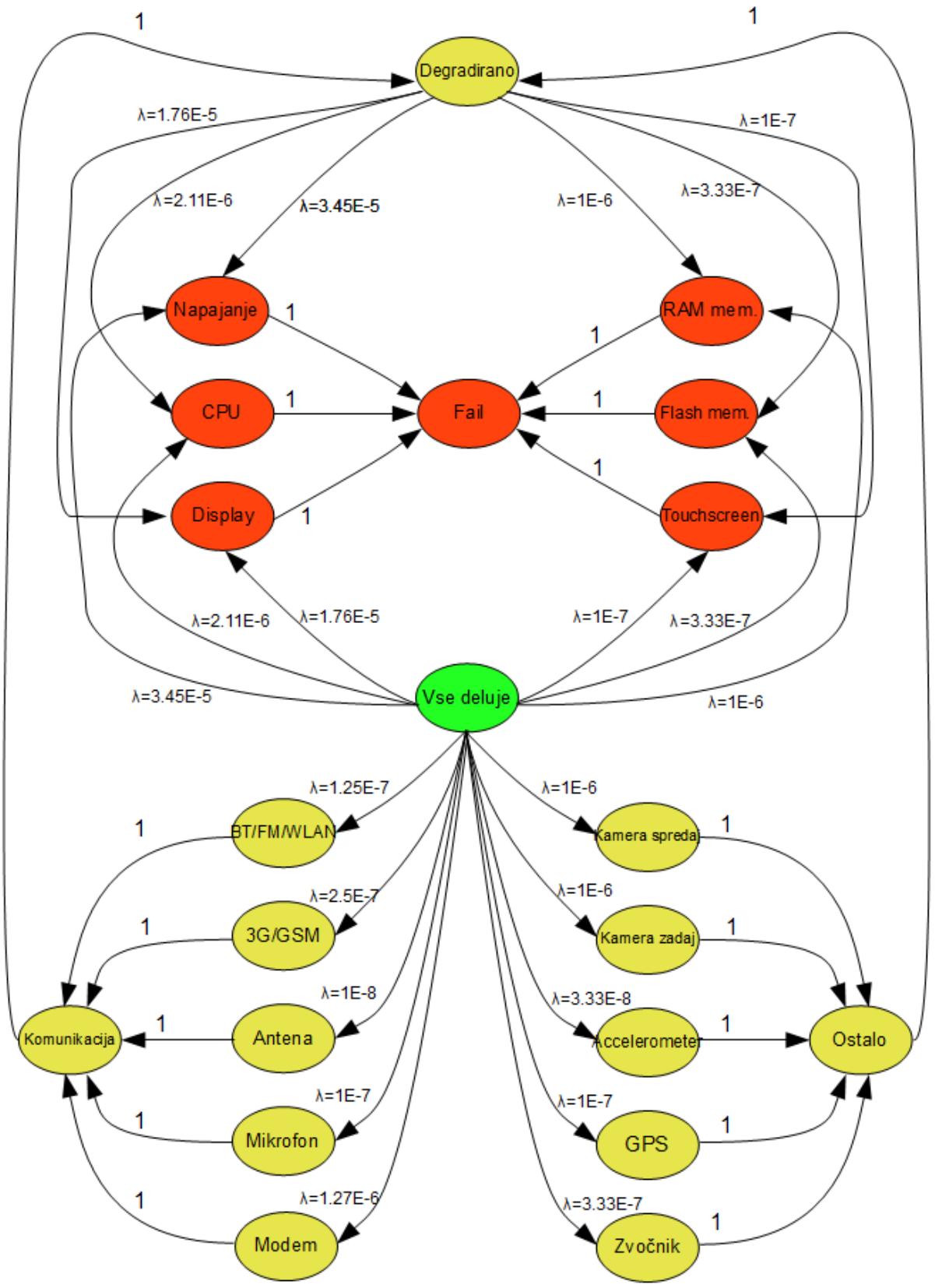
- napajanje sistema
- centralna procesna enota-CPE
- zaslon
- zaslon na občutljiv na dotik
- Flash pomnilnik
- RAM pomnilnik

Potem tu nastopi druga skupina komponent, kjer ob odpovedi le-teh sistem še vedno deluje, ampak z nižjo funkcionalnostjo. Tem stanju sistema pravimo degradirano stanje oz. v angleščini "Degraded". Te bi lahko razdelili na dve skupini:

1. Komponente, ki služijo za komunikacijo:
   - BT, FM, LAN
   - 3G, GSM
   - antena
   - mikrofon

2. Ostale komponente:
   - kamera spredaj
   - kamera zadaj
   - senzor občutljiv na gibanje naprave
   - GPS
   - Zvočnik

Na naslednji strani je prikazan diagram prehajanja stanj narejen v razvojnem okolju za markovske analize. Kot lahko vidimo smo razporedili nekatere komponente v skupine. To smo naredili zato, da smo se izognili prevelikemu številu povezav med komponentami. Model je še vedno matematično pravilen, saj so prehodi med skupinami hipni oz. $\lambda = 1$. Torej prehodi med dodatnimi skupinami verjetnostno ne vpliva na pogojne verjetnosti prehodov med komponentami.



Slika 6.1: Diagram prehajanja stanj celotnega sistema

6.2.2. Intenzivnosti odpovedovanja posameznih komponent

Spodaj sta priloženi tabeli z intenzivnostmi odpovedovanja posameznih komponent, ki sestavljajo naš sistem. Za nekatere izmed komponent je intenzivnosti izračunala skupina sošolcev po MIL standardu. Za ostale komponente je naša skupina dobila podatke na internetnih forumih. Podatki so se precej razlikovali, včasih tudi za dva velikostna razreda. V takšnih primerih smo za intenzivnost vzeli aritmetično sredino.

Komponenta	λ
BT/FM/WLAN	1.25E-7
3G/GSM	2.50E-7
HSDPA/HSUPA/EDGE modem	1.27E-6
Acelerometer	3.33E-8
GPS	1.00E-7
Zvocnik	3.33E-7
Napajanje	3.45E-5
CPU	2.11E-6
Display	1.76E-5
Touchscreen krmilnik	1.00E-7
Flash	3.33E-7

Tabela 6.1: Tabela intenzivnosti izračunanih po MIL standardu

Komponenta	λ
Antena	1.00E-008
Mikrofon	1.00E-007
Kamera spredaj	1.00E-6
Kamera zadaj	1.00E-6
RAM	1.00E-006

Tabela 6.2: Tabela intenzivnosti pobranih iz internetnih forumov

Za opombo moram omeniti, da smo za intenzivnost odpovedovanja kamere dobili podatke na internetu izmed 1.00E-5 in 1.00E-7. Zato smo vzeli približno vrednost 1E-6. Odpoved antene je največ internetnih virov navajalo kot približno 1E-8. Za RAM-e, ki so vgrajeni v čip smo vzeli približno 1E-6. Odločili smo se na snovi njihove izpostavljenosti.

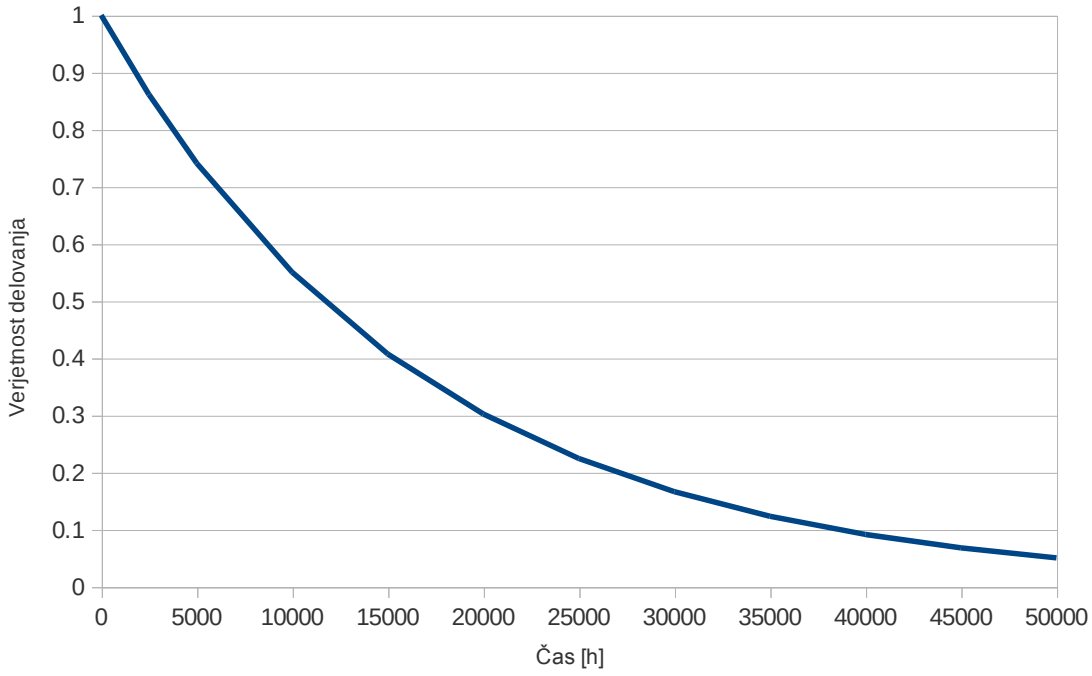
6.2.3. Rezultati testiranja

Zdaj je pred nami najpomembnejši del tega projekta. To so rezultati testiranja zanesljivosti delovanja celotnega sistema z uporabo markovskega modela. Torej v predhodnem podpoglavju smo sestavili model našega sistema v določenem razvojnem okolju in v tem koraku smo pognali simulacijo obnašanja sistema za 50.000 h kar znaša približno 6 let. Predpostavili smo, da se v večini primerov izdelek kot je naš zamenja za novega, saj je verjetnost delovanja majhna in zaradi želje po zamenjavi izdelka za novega, tehnološko naprednejšega. Zato je zanesljivost delovanja od te časovne točke naprej nezanimiva, saj sistema takrat navadno ne uporabljamo več. Časovni korak analize, ki smo ga izbrali je 5000 h, kar ustreza približno 7 mesecem. Na takšen način smatramo, da je obnašanje sistema opisano dovolj natančno.

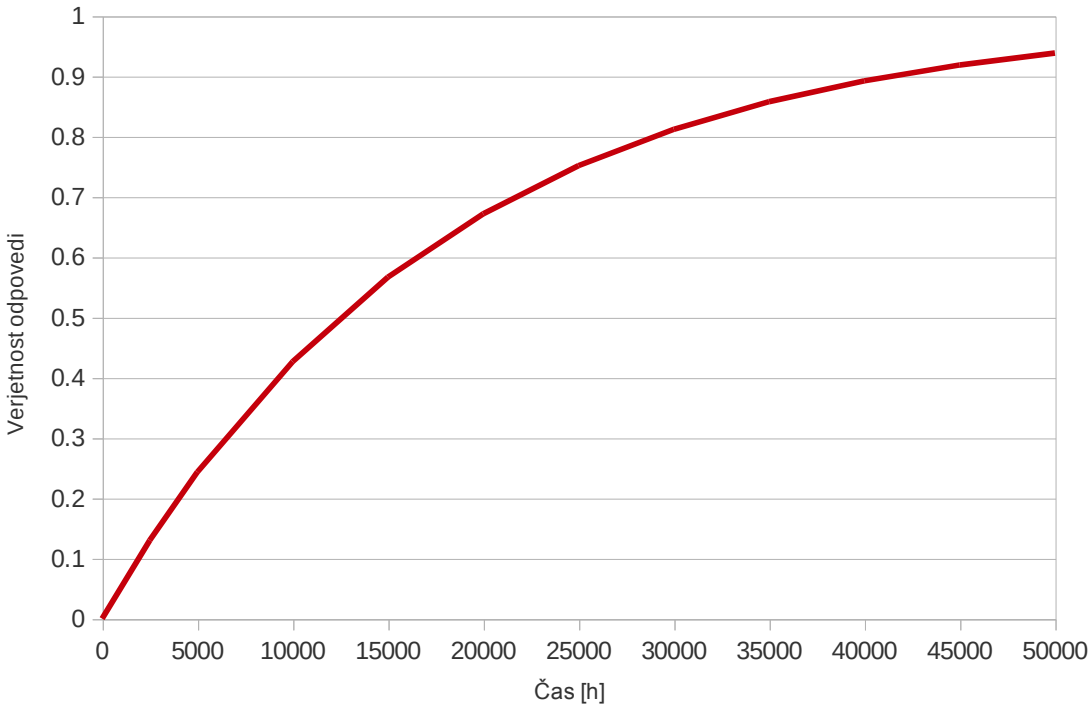
Čas [h]	Delujoče stanje	Degradirano st.	Nedelujoče st.
2500	0.86100	0.00913	0.12987
5000	0.74100	0.01580	0.24320
10000	0.55000	0.02370	0.42630
15000	0.40700	0.02660	0.56640
20000	0.30200	0.02660	0.67140
25000	0.22400	0.02490	0.75140
30000	0.16600	0.02240	0.81160
35000	0.12300	0.01960	0.85740
40000	0.09120	0.01680	0.89200
45000	0.06760	0.01410	0.91830
50000	0.05010	0.01180	0.93810

Tabela 6.3: Rezultati zanesljivosti celotnega sistema po 50.000 urah (cca 6 let)

Zelo zanimiv rezultat iz tabele je, da je verjetnost popolnega delovanja celotnega sistema po približno 6 letih uporabe samo 5%. Še bolj zanimiv pa je grafični prikaz eksponentnega padanja zanesljivosti skozi čas in naraščanje verjetnosti za odpoved sistema. Grafi, ki to ponazarjajo za vsa možna stanja sistema (delujoče, degradirano in nedelujoče) se nahajajo spodaj.

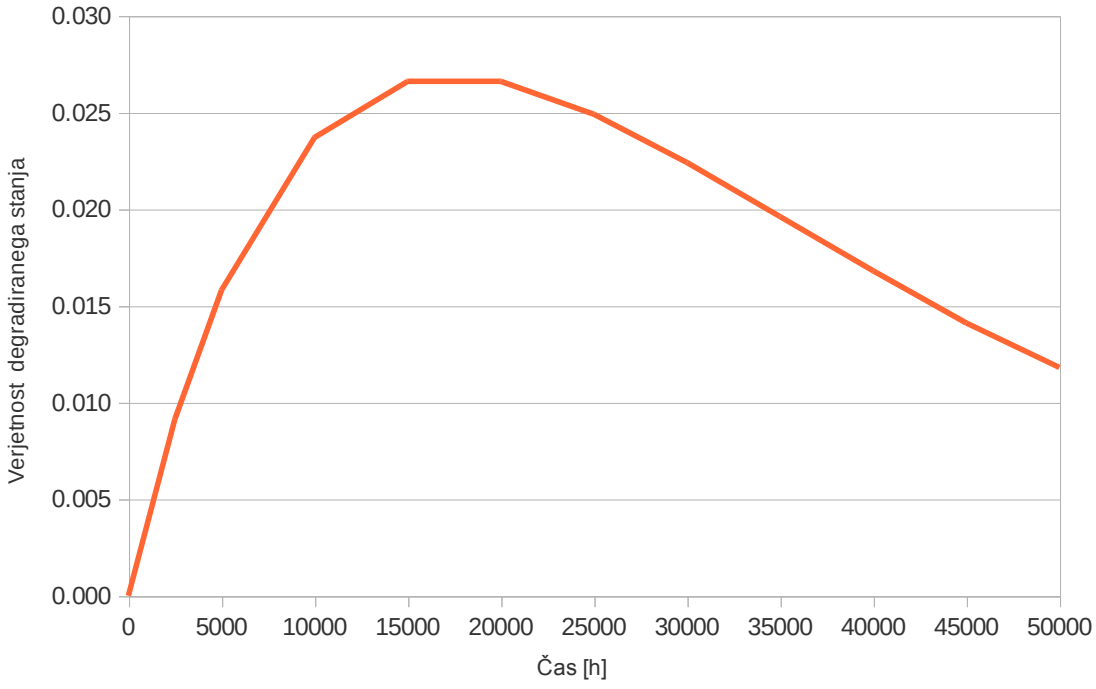


Graf 6.1: Upadanje zanesljivosti delovanja celotnega sistema



Graf 6.2: Rast verjetnosti odpovedi celotnega sistema

Na spodnjem grafu lahko vidimo zanimiv pojav. To je rast verjetnosti degradiranega stanja do neke časovne točke, kjer ta doseže maksimum, potem pa začne verjetnost upadati. To se zgodi zato, ker začne sistem prehajati iz degradiranega stanja v nedelujoče stanje.



Graf 6.3: Rast verjetnosti obstoja sistema v degradiranem stanju

6.3. Diskusija

Tukaj bomo posvetili nekaj besedila načinu postavitve modeliranja za markovsko analizo in kasnejšo simulacijo. Najprej smo uporabili orodje Isograph, katerega glavna lastnost je natančen izračun zanesljivosti delovanja skozi čas obratovanja sistema. Glavni problem tega orodja pomanjkanje možnosti za shranjevanje nedokončanega modela. Zato smo prešli na orodje CARMS, kjer smo postavili celoten sistem in pognali simulacijo. Na koncu smo isti model prenesli na Isograph in še enkrat pognali simulacijo za primerjavo natančnosti izračuna. Izračuni obeh programov so bili skladni. Tretje orodje, s katerim smo se na začetku ukvarjali je bilo Relex, ampak smo zaradi pomanjkanja funkcionalnosti okrnjene različice uporabo opustili.

Druga stvar, ki jo moram poudariti je natančnost same analize. Podatke o zanesljivosti komponent je s strani proizvajalcev sistema nemogoče dobiti, zato smo te vrednosti morali po standardih izračunati sami. Ti izračuni pa so približni in zato so tudi rezultati analize, kjer smo te izračune uporabljali kot osnovo samo približni, ampak še vedno precej realistični.

7. Viri

[1.1] http://st2.gsmarena.com/vv/pics/samsung/samsung-galaxy-tab-ofic-1.jpg

[1.2] http://www.ifixit.com/Teardown/Samsung-Galaxy-Tab-Teardown/4103/2

[2.1] MIL-HDBK-217F, Reliability Prediction of Electronic Equipment, 1991. Notice 1(1992) and Notice 2(1995).

[2.2] http://www2.electronicproducts.com/Galaxy_Tab_Gt_P1000_-whatsinside_text-101.aspx

[2.3] I. Vizec, Analiza standardov za določanje zanesljivosti elektronskih komponent, Diplomsko delo, 2010

[2.4] http://www.ptc.com/products/windchill/prediction

[2.5] http://www.boardcon.com/download/S5PC100_UM_REV101.pdf

[2.6] http://wenku.baidu.com/view/b414c4db50e2524de5187e0e.html

[2.7] http://www.infineon.com/dgdl/X-GOLD+616.pdf?folderId=db3a304312fcb1bc0113000c158f0004&fileId=db3a30431ed1d7b2011f5bee88ef75eb

[2.8] IEC TR 6280: Reliability data handbook – Universal model for reliability prediction of electronics components, PCBs and equipment

[2.9] http://en.wikipedia.org/wiki/Lithium-ion_battery

[4.1] http://en.wikipedia.org/wiki/Fault_tree_analysis

[4.2] http://eprints.fri.uni-lj.si/258/1/%C5%BDnidar%C5%A1i%C4%8D_G_VS.pdf

[4.3] http://lrss.fri.uni-lj.si/sl/teaching/rzd/lectures/6_poglavje.pdf

[5.1] Wikipedia. Failure mode and effects analysis. http://en.wikipedia.org/wiki/Failure_mode_and_effects_analysis. 18.5.2011

[5.2] Wikipedia. Failure mode, effects, and criticality analysis. http://en.wikipedia.org/wiki/FMECA. 18.5.2011

[6.1] A.A. Markov. "Rasprostranenie zakona bol'shih chisel na velichiny, zavisyaschie drug ot druga". *Izvestiya Fiziko-matematicheskogo obschestva pri Kazanskom universitete*, 2-ya seriya, tom 15, pp. 135–156, 1906.

[6.2] CARMS orodje: http://www.tc.umn.edu/~puk/carms.htm

[6.3] Isograph reliability workbench: http://www.isograph-software.com/download.php?src=isoswr, http://www.plant-maintenance.com/freestuff/1004.shtml