

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Iztok Starc

**Zagotavljanje varnosti za okolja
omrežnih sistemov radiofrekvenčne
identifikacije**

MAGISTRSKO DELO

Ljubljana, 2011

Št.: 140-MAG-RI/2011

Datum: 27. 05. 2011



Iztok STARC, univ. dipl. inž. el

Ljubljana

Fakulteta za računalništvo in informatiko Univerze v Ljubljani izdaja naslednjo magistrsko nalogo

Naslov naloge: **Zagotavljanje varnosti za okolja omrežnih sistemov radio-frekvenčne identifikacije**

Security assurance in networked radio-frequency identification system environment

Tematika naloge:

Definirajte in izdelajte metodo za zagotavljanje varnosti okolij omrežnih sistemov RFID, tako da bodo upoštevane trenutne in bodoče grožnje, ki bi izkoristile ranljivosti značk RFID. Zagotavljanje varnosti okolij naj poteka s testiranjem varnosti značke RFID povezane v omrežni sistem RFID. Metoda naj bo osnovana na programirljivi matriki logičnih vrat FPGA in strojnem opisnem jeziku VHDL za opis obnašanja značke RFID, tako da bo izkoriščena ekvivalenca preslikave tehnološko nevtralnega opisa značke RFID na FPGA in aplikacijsko specifična vezja ASIC. Emulacija značke RFID povezane v omrežni sistem RFID naj omogoča testiranje funkcionalnosti varnostnih protiukrepov, testiranje varnosti na podlagi prepoznanih tveganj pri varovanju informacij ipenetracijsko testiranje, vse pred operativno uporabo sistema RFID.

Mentor:

prof. dr. Denis Trček



Dekan:

prof. dr. Nikolaj Zimic

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Iztok Starc, univ. dipl. inž. el.

**Zagotavljanje varnosti za okolja
omrežnih sistemov radiofrekvenčne
identifikacije**

MAGISTRSKO DELO

Mentor: prof. dr. Denis Trček

Ljubljana, 2011

Namesto te strani **vstavite** original izdane teme magistrskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!

IZJAVA O AVTORSTVU

magistrskega dela

Spodaj podpisani Iztok Starc,

z vpisno številko 63060515,

sem avtor magistrskega dela z naslovom:

Zagotavljanje varnosti za okolja omrežnih sistemov radio-frekvenčne identifikacije

S svojim podpisom zagotavljam, da:

- sem magistrsko delo izdelal samostojno pod mentorstvom prof. dr. Denisa Trčka
- so elektronska oblika magistrskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike magistrskega dela v zbirki "Dela FRI".

V Ljubljani, dne 14. 9. 2011

Podpis avtorja:

ZAHVALA

Leta 2008 sem v okviru podiplomskega študija računalništva in informatike in kot diplomant elektrotehnike spoznal tehnologijo radio-frekvenčne identifikacije (RFID). Navdušila me je. Spoznal sem številne praktične primere uporabe in vrsto neizkoriščenih možnosti, ki še čakajo na svoj trenutek. Poleg tega sem odkril tudi temno plat uporabe tehnologije in njen vpliv na varnost in zasebnost slehernega posameznika v družbi, združb in vlad. Odkril sem tudi številne primere izkoriščanja ranljivosti tehnologije v nelegitimne namene. Presenečen sem bil nad višino nastale škode. V okviru raziskovalnega dela me je zanimalo, kako izboljšati obvladovanje tveganja pri varovanju informacij za okolja omrežnih sistemov RFID za bolj varno tehnologijo in za boljši svet. Magistrsko delo je plod tega prizadevanja.

Zahvaljujem se mentorju, prof. dr. Denisu Trčku, da je to idejo podprl in jo podpiral vse do njene realizacije. Mentorju se prav tako zahvaljujem, da mi je omogočil osnovne pogoje za delo in prosto pot kreativnosti.

Prof. dr. Burkhardu Stillerju iz Univerze v Zürichu se zahvaljujem za koristne nasvete pri načrtovanju tehničnega izdelka.

Za sproščeno delovno vzdušje, tudi v času večjih obremenitev, se zahvaljujem sedanjim in bivšim sodelavcem Laboratorija za e-medije Blažu, Damjanu, Davidu, Evi, Gašperju, Mateju in Iztoku. Mateji Kosi se zahvaljujem za lektoriranje magistrskega dela in pomoči pri gojenju slovenskega jezika. Suzani Ščavničar se zahvaljujem za lektoriranje angleškega povzetka. Popravke sem upošteval v največji možni meri.

Hvala moji družini, sestri Urški, mami Katarini in očetu Iztoku, da so ves čas verjeli vame. Brez njihove izdatne moralne, čustvene in finančne podpore bi bili vsi vloženi naporji najbrž zaman.

Zelo posebna zahvala je namenjena Marini za njeno ljubezen in potrpežljivost ter za razumevanje za vse večere in konce tednov, ki sem jih preživel pred računalniškim zaslonom med izdelavo tega dela.

V Murski Soboti, dne 3. septembra 2011

Iztok Starc

Per aspera ad astra.
“Skozi trnje do zvezd.”

Povzetek	ix
Abstract	xi
1 Uvod	1
2 Pregled področja	3
2.1 Zgodovinski pogled in prihodnost	3
2.2 Opis tehnologije	4
2.3 Pomen kakovosti sistema	6
2.3.1 Karakteristike kakovosti	6
2.3.2 Odvisnosti karakteristik	7
2.4 Varnost	8
2.5 Grožnje	9
2.5.1 Od groženj do izkoriščanja ranljivosti	9
2.5.2 Grožnje omrežnih sistemov RFID	9
2.6 Protiukrepi	12
2.7 Obvladovanje tveganja	12
2.8 Zagotavljanje varnosti	14

2.8.1	Tipologija metod zagotavljanja varnosti	15
2.8.2	Zahteve orodij za preizkušanje varnosti	15
2.8.3	Načrtovanje aktivnosti preizkušanja varnosti	15
2.9	Cilj magistrske naloge	16
3	Načrt orodja	17
3.1	Analiza zahtev	17
3.1.1	Funkcionalne zahteve	18
3.1.2	Omejitve pri realizaciji značke	18
3.1.3	Omejitve aparaturnega priklopa značke	20
3.1.4	Omejitve pri realizaciji delovnega mesta preizkuševalca	20
3.1.5	Optimizacija poteka preizkušanja	21
3.2	Arhitektura orodja	23
3.2.1	Kontekst demonstracijskega sistema	23
3.2.2	Emulacijsko okolje	23
3.2.3	Komunikacijski tunel značke	30
3.2.4	Podporna oprema komunikacijskega tunela značke	33
3.2.5	Komunikacijski tunel čitalca	36
3.2.6	Orodje za preizkušanje varnosti	38
3.2.7	Čitalec	42
3.2.8	Zaledni del sistema	45
3.3	Postopek preizkušanja varnosti	51
3.4	Metoda za zagotavljanje varnosti	61
3.4.1	Proces obvladovanja tveganja	61
3.4.2	Prepoznavanje ranljivosti	63
3.4.3	Pozitivni učinki uporabe metode	64
4	Analiza	67

4.1	Karakteristike orodja	68
4.1.1	Transparentnost	68
4.1.2	Prilagodljivost	70
4.1.3	Zanesljivost	70
4.1.4	Zmogljivost	71
4.1.5	Razširljivost	72
4.2	Omrežna arhitektura orodja	72
4.3	Omejitve orodja	73
4.4	Dinamika kakovosti sistema	74
4.4.1	Model	75
4.4.2	Simulacije	75
4.4.3	Nova ranljivost pri aktivni uporabi sistema	79
4.4.4	Nabava ranljivega sistema	86
4.4.5	Nabava ranljivega sistema z nastankom nove ranljivosti	94
5	Zaključek	103
5.1	Prispevki	104
5.2	Nadaljnje delo	105
A	Zahteve za načrtovanje in razvoj orodja	107
B	Postopek načrtovanja, razvoja in namestitve sistema MicroBlaze	115
B.1	Postopek definicije strojne opreme	115
B.2	Postopek gradnje strojne opreme	127
B.3	Postopek gradnje osnovnih programskih orodij	128
B.4	Vmesni preizkus strojne opreme	130
B.5	Povzetek poglavja	134
C	Postopek načrtovanja, razvoja in namestitve operacijskega sistema Petalinux	135

C.1	Priprava okolja za definicijo in gradnjo operacijskega sistema uClinux . . .	135
C.2	Postopek sinhronizacije nastavitve strojne opreme in okolja Petalinux . . .	137
C.3	Gradnja operacijskega sistema	139
C.4	Postopek namestitve slike operacijskega in datotečnega sistema ter poskusni zagon sistema	147
C.5	Preizkus omrežne povezljivosti	151
C.6	Preizkus naprave GPIO	154
C.7	Povzetek poglavja	156
D	Postopek načrtovanja, razvoja in namestitve testne aplikacije v datotečni sistem PetaLinux	157
D.1	Opis testne aplikacije	157
D.2	Gradnja in namestitve teste aplikacije	157
D.3	Gradnja odjemalca za testno aplikacijo	162
D.4	Preizkus testne aplikacije	165
D.5	Povzetek poglavja	166
E	Izpisi izvirne kode	167
E.1	Primer opisa značke	167
E.2	Primer opisa protokolnega analizatorja	169
E.3	Izvedba vmesnika čitalca	170
F	Seznam strojne in programske opreme	173
F.1	Strojna oprema za razvoj emulacijskega sistema	173
F.2	Programska oprema za razvoj emulacijskega sistema	173
F.3	Strojna oprema emulacijskega sistema	173
F.4	Programska oprema emulacijskega sistema	174
G	Simulacijski model	175
H	Izvorna koda	183

Slike	185
Tabele	192
Izpisi	194
Literatura	197
Ostali viri	200

Seznam kratic in simbolov

ACID	Atomicity, Consistency, Isolation, Durability
ASIC	Application Specific Integrated Circuit
DDR	Double Data Rate
DNS	Domain Name System
EDK	Embedded Development Kit
EPC	Electronic Product Code
FIFO	First In, First Out
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPIC	Intellectual Property InterConnect
IPIF	Intellectual Property InterFace
ISO	International Organization for Standardization
ITU	International Telecommunication Union

JTAG	Joint Test Action Group
MAC	Media Access Control
ONC	Open Network Computing
OSI	Open Systems Interconnection
PCI	Peripheral Component Interconnect
POSIX	Portable Operating System Interface for uniX
RAM	Random Access Memory
RFID	Radio Frequency IDentification
RPC	Remote Procedure Call
RPCL	Remote Procedure Call Language
RS-232	Recommended Standard 232
RTL	Register Transfer Level
SDRAM	Synchronous Dynamic Random Access Memory
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
VHDL	Very-high-speed integrated circuits Hardware Description Language
WSDL	Web Services Description Language
XDR	eXternal Data Representation
XML	eXtensible Markup Language

Povzetek

Radiofrekvenčna identifikacija (RFID) je informacijsko-komunikacijska tehnologija za označevanje in brezžično identifikacijo predmetov. Tipični sistem RFID sestavljajo značka, čitalec in zaledni del sistema s podatkovno bazo. Ti sistemi so dandanes prisotni tako v panogah privatnega sektorja kot tudi v javnem sektorju. Trend postavitve sistemov RFID narašča, saj visok nivo avtomatizacije dela združbam omogoča priložnost za reorganizacijo in prilagoditev njihovih poslovnih procesov za zmanjšanje stroškov, povečanje prihodkov, izboljšanje ravni poslovnih storitev in povečanje števila stalnih strank. Temu trendu pa sledi tudi odvisnost združb od kakovosti delovanja sistemov RFID.

Zagotavljanje varnosti in zasebnosti lahko razumemo tudi kot enega od vidikov kakovosti sistemov RFID. Struktura sistemov RFID je kompleksna, tehnološko heterogena in viri sistema so omrežno povezani na internet. Naštete lastnosti omogočajo hekerjem oddaljeno in avtomatizirano izkoriščanje ranljivosti izpostavljenih sistemov. Poleg tega raznoliki konteksti uporabe sistemov vplivajo na kompleksnost procesa obvladovanja tveganja pri varovanju informacij in zahtevnost načrtovanja in razvoja sistema za upravljanja varovanja informacij. Trenutno obvladovanje tveganja je reaktivno, saj kasni glede na incidente. Zaradi tega napadi na sisteme RFID povzročijo veliko škode in nezadovoljstvo deležnikov. Pomisleki deležnikov o varnosti in zasebnosti so zato ena od glavnih ovir pri uvajanju tehnologije RFID.

V magistrskem delu sta zasnovana in razvita metoda in orodje za zagotavljanje varnosti za okolja omrežnih sistemov RFID. Orodje je uporabljeno v okviru metode in omogoča simulacijo napadov na sistem RFID. Napadi so izbrani na podlagi prepoznanih groženj in omogočajo preizkuševalcu popoln nadzor nad izvajanjem logičnih operacij značke in nad komunikacijo med značko in čitalcem. Orodje je stroškovno učinkovito, saj sta značka in del orodja realizirana z matriko programirljivih logičnih vrat FPGA, ostali deli orodja pa z izdelki široke potrošnje. Orodje je zaradi nizkih stroškov dostopno širšemu krogu raziskovalcev. Orodje omogoča testiranje funkcionalnosti varnostnih protiukrepov značke, testiranje varnosti sistema na podlagi prepoznanih tveganj in penetracijsko testiranje sistema. Metoda za zagotavljanje varnosti je poravnana s procesom obvladovanja tveganja

pri varovanju informacij ISO/IEC 27005:2008. Uporaba orodja in metode v okviru svojih zmožnosti omogoča spremembo pristopa obvladovanja tveganja z reaktivnega, ki sledi posledicam in predstavlja trenutno stanje, na proaktivno, ki prehiteva dogodke incidentov in jih preprečuje.

Ključne besede: *varnost, zasebnost, obvladovanje tveganja pri varovanju informacij, zagotavljanje varnosti, RFID, omrežni sistemi RFID, orodje za testiranje varnosti, FPGA.*

Abstract

Radio frequency identification is an information and communication technology for wireless identification and object labelling. A typical RFID system consists of a tag, a reader and a back-end system connected to a database. Nowadays these systems are present in the private sector industries as well as in the public sector. The RFID system deployment is increasing as high level of automation allows companies the opportunity to re-organize and adapt their business processes to reduce costs, increase revenue, improve services and increase the number of regular customers. However, this trend is also followed by increasing dependence on security, which is one of the system's quality most notable characteristic from the stakeholder's point of view.

The technological structure of RFID system is complex, distributed, whereas components are heterogeneous and they are interconnected through internet. This allows hackers to mount remote and automated attacks by exploiting vulnerabilities in exposed systems and may cause unrecoverable damage to business. Further, the complexity of information security risk management process and difficulty in implementing an information security management system are negatively affected by the system's complexity and its diverse context of use. The current information security risk management is basically reactive as it is lagging behind incidents. This strategy has proven inefficient and the number of dissatisfied stakeholders is increasing. Stakeholders' concerns regarding security and privacy are one of the main barriers to RFID technology adoption.

The Master's thesis hereof defines and develops a security assurance method and a tool for provisioning security. This tool is used in conjunction with the proposed method and enables simulated cyber attacks on a generic networked RFID system. The cyber attack selection criteria are based upon identified threats and allow the tester to have a complete RFID tag execution control and can overhear, intercept and synthesize any message on the communication channel between the tag and the reader as well. The RFID tag and the tool stub are implemented in the low-cost field programmable gate array FPGA. The second part of the tool is implemented in terms of using commodity hardware and software. This cost effective implementation makes the tool affordable to the

security research community aimed at performing functional security testing, risk-based security testing and penetration testing. Furthermore, the security assurance method is aligned with the ISO/IEC 27005:2008 information security risk management process. The aforesaid gives companies the opportunity to change the information security management strategy from reactive to proactive and accordingly increases the odds of a timely risk identification and thus prevents the occurrence of incidents.

Keywords: *security, privacy, information security risk management, security assurance, RFID, networked RFID systems, security tool, FPGA.*

RADIOFREKVENČA IDENTIFIKACIJA (RFID) je informacijsko-komunikacijska tehnologija za označevanje in brezžično identifikacijo predmetov.

Tipičen sistem RFID sestavljajo značka, čitalec in zaledni del sistema s podatkovno bazo. Končnemu uporabniku so na voljo analitične in upravljalске storitve, ki za svojo izvedbo zahtevajo storitev identifikacije predmetov. Le-ta je podatkovna storitev, če so zahteve za hitrost in izvedbo dovolj ohlapne, v nasprotnem primeru pa storitev v realnem času. Zaradi gibljivosti značk po prostoru se uvršča tudi med mobilne storitve. Namen uporabe analitičnih in upravljalških storitev po tipologiji NIST SP 800-98 [1] je sledenje predmetov (določitev lokacije predmeta), odkrivanje ponaredkov (preverjanje izvora predmeta), preverjanje ujemanja skupine predmetov (zagotavljanje določenih lastnosti skupine predmetov), vodenje procesov (uporaba dogodkov predmetov za odločanje), nadzor dostopa (namen identifikacije je izvedba postopka overjanja osebe), samodejno plačevanje storitev (izvedba finančne transakcije) in vodenje oskrbne verige (določitev prisotnosti predmeta).

Raznolikost primerov uporabe tehnologije RFID omogoča združbam priložnost za reorganizacijo in prilagoditev njihovih poslovnih procesov za zmanjšanje stroškov, povečanje prihodkov, izboljšanje ravni poslovnih storitev in povečanje števila stalnih strank. To so ključni poslovni dejavniki za donosnost naložbe v novi sistem. Poleg tega so moderni sistemi omreženi in zaradi tega tudi razširljivi. Ta lastnost omogoča ekonomijo obsega in predstavlja priložnost za dodatno znižanje stroškov pri izvajanju storitev. Pri povezovanju združljivih sistemov RFID različnih združb so doseženi še dodatni sinergijski učinki. Ti sistemi so dandanes prisotni tako v panogah zasebnega sektorja kot tudi v javnem sektorju, na primer v naslednjih vejah industrije: avtomobilistična, bančna, biotehno-loška, elektrotehnična, informacijska in komunikacijska, kemijska, letalska in vesoljska, obrambna, pomorska, prodajna, tekstilno predelovalna, zdravstvena in živilska [2].

Po sklepu tržne raziskave IDTechEx iz leta 2010 [56] trend postavitve sistemov narašča, saj združbe pozitivno ocenjujejo, da se vložek v sistem povrne na srednji rok, ne glede na nazadovanje gospodarske rasti v letih 2008–2011. Druga tržna raziskava ABIResearch iz leta 2009 [57] predvideva 15-odstotno rast na letni ravni za obdobje petih let pri upoštevanju nazadovanja gospodarske rasti.

Iz obeh tržnih raziskav lahko sklepamo, da bo vedno večje število poslovnih sistemov in njihovih poslovnih procesov odvisnih od storitev sistema RFID in njihove kakovosti. Visoka raven avtomatizma, kompleksnost sistemov in njihova omrežna povezanost pa otežujejo zagotavljanje kakovosti sistemov [3, str. 94]. Še več, takšni sistemi pogosto vsebujejo ranljivosti, ki jih je mogoče izrabiti na nelegitimen način. Novi načini izrabe ranljivosti predstavljajo poslovno priložnost za kibernetško kriminaliteto. Zgoraj navedene lastnosti sistema so še posebej privlačne zaradi možnosti oddaljenega in avtomatiziranega napada[†].

Zaradi interneta se kriminalne skupnosti enostavno orgnizirajo in povezujejo, kar omogoča trivialno razpečevanje orodij in znanja za izvajanje kriminalne dejavnosti. Zaradi avtomatiziranega in oddaljenega napada je potencialna škoda lahko zelo velika. Po drugi strani je odkrivanje in prijetje storilcev oteženo.

V magistrskem delu se bomo osredotočili na zagotavljanje varnosti in zasebnosti sistemov RFID. Sledeč poročilu Evropske komisije so pomisleki glede varnosti in zasebnost glavne ovire pri uvajanju sistemov RFID [5], poleg visokih stroškov postavitve sistema in pomanjkanja globalnih standardov za interoperabilnost sistemov. Legitimni uporabniki sistemov se soočajo z različnimi tveganji pri varovanju varnosti in zasebnosti, o čemer priča zgodovina napadov na sisteme RFID. Ta tveganja izhajajo iz groženj, ki so specifična za vsak kontekst uporabe sistema posebej. Prepoznana tveganja je potrebno učinkovito obvladovati, tako da je strošek obvladovanja tveganja manjši od stroška neodpravljenе ranljivosti in je cenovno in časovno učinkovit.

V magistrskem delu nas bo zanimalo kako izboljšati zagotavljanje varnosti za okolja omrežnih sistemov RFID in prispevati k bolj varni tehnologiji za uporabnike in boljši svet.

Organizacija magistrskega dela

V drugem poglavju bomo obravnavali zgodovinske in tehnične vidike sistemov RFID. Izpostavili bomo problem obvladovanja tveganja in zagotavljanja varnosti, ki bo rdeča nit magistrske naloge. V tretjem poglavju bomo analizirali zahteve za razvoj orodja. Na podlagi specifikacije zahtev bo predstavljena arhitektura orodja. Predstavljen bo tudi postopek preizkušanja varnosti in umestitev postopka v metodo obvladovanja tveganja. V četrtem poglavju bo sledila analiza orodja in metode, pri čemer nas bodo zanimale karakteristike orodja, omrežna arhitektura, omejitve orodja in vpliv uporabe orodja na dinamiko kakovosti sistema. V zaključku bodo podani sklepne ugotovitve, prispevki in nastavki za nadaljnje delo.

[†]Kritična infrastruktura iranskega jedrskega programa je bila napadena z zlonamerno kodo Stuxnet [4]. Po domnevah Irana naj bi bila zlonamerna koda izdelana v Združenih državah Amerike in Izraelu, vendar sta to trditve obtoženi državi zanikali.

2.1 Zgodovinski pogled in prihodnost

BREZZIČNA TEHNOLOGIJA ima temelje v vrsti znanstvenih odkritij 19. in 20. stoletja. Faradayevemu odkritju elektromagnetne indukcije (1846) je sledila Maxwellova poenotena matematična teorija elektromagnetizma (1864), ki je predpostavila obstoj elektromagnetnega valovanja, torej valovanja, ki se v praznem prostoru razširja s svetlobno hitrostjo in ob pravi valovni dolžini zavzame obliko vidne svetlobe. Te napovedi so bile prvič demonstrirane šele 23 let po objavi teorije s Hertzovo napravo (1887). Ta naprava je s pomočjo dipolne antene sevala in sprejemala radijske valove. Popov, Marconi in ostali so v nadaljevanju izpopolnjevali delo Hertza. Leta 1896 se je pojavila radiotelegrafija in nekoliko kasneje še radio (1906). Domet storitev prenosa podatkov in zvoka se je povečal z inovacijami na področju elektronike, tj. s Flemingovo diodo kot usmernikom (1904) in z De Forrestovo triodo kot ojačevalcem signala (1906). Radio kot brezžično mobilno storitev je do konca prve svetovne vojne pretežno uporabljala vojska. Po vojni je radio doživel razcvet in povzročil pravo revolucijo na področju množičnih občil. Razvoj radia se je nadaljeval v smeri vse boljše kakovosti reprodukcije zvoka in učinkovitosti izkoriščenja komunikacijskega medija.

Radijske valove je mogoče uporabiti tudi na druge načine. Leta 1904 je Hülsmeier patentiral napravo, ki je izkoriščala odbojnost radijskih valov za preprečevanje nesreč na morju. Ta pojav se uvršča med splošni pojav odboja elektromagnetnega vala, kamor sodi tudi odboj svetlobe in ga človeštvo na primer uporablja pred ogledalom med jutranjimi opravili. Watt je leta 1935 inoviral napravo radar, ki je izkoriščala odbojnost radijskih valov od zračnih plovil. Položaj, smer in hitrost gibanja oddaljenih predmetov sklepamo na podlagi odboja in to ne neodvisno od vremenskih motenj ter s tem povezane omejene vidljivosti. Ta strateška prednost je bila najprej izkoriščena za vojaške namene, po drugi svetovni vojni pa je doživela še široko uporabo v civilnem letalstvu.

Radarske slike vse do nedavnega ni bilo mogoče uporabiti za razlikovanje med različnimi tipi zavezniških in sovražnikovih letal [6]. Zato so zračne sile osi med drugo svetovno vojno razvile pasivno metodo identifikacije z rotiranjem letala okrog svoje osi. Na ta način se

je spremenila odbojna površina, ki se je manifestirala kot utripanje pike na radarski sliki. Zračne sile zaveznikov so po drugi strani razvile aktivno metodo identifikacije z uporabo transponderja. Ta je začel oddajati radijski signal, takoj ko je zaznal prisotnost radarja.

To je primer ranljivosti, ki so jo odkrile tudi sile osi in zaradi posledic katere so bila izgubljena tudi človeška življenja. Kot se je ugotovilo kasneje, je bilo tveganje pri uporabi transponderja mogoče zmanjšati s preprostim protiukrepom. Napravo je bilo potrebno izklopiti v sovražnikovem zračnem prostoru [7].

Radarska tehnologija in uporaba le-te za preprosti prenos podatkov je vplivala na idejno zasnovo radiofrekvenčne identifikacije (1948) z objavo prispevka na temo komunikacij s povratnim sipanjem moči [8]. Objavljene ideje so bile pred svojim časom predvsem zaradi omejenih tehničnih zmožnosti za gradnjo naprav. Tako je bil prvi patent, ki opisuje primer uporabe tehnologije, podeljen šele leta 1973, in sicer za napravo za avtomatizacijo cestninjenja [9]. Napredki, najprej na področju elektrotehnike, nato pa računalništva in informatike, so omogočili razcvet in uporabo tehnologije v vodilnih industrijskih panogah.

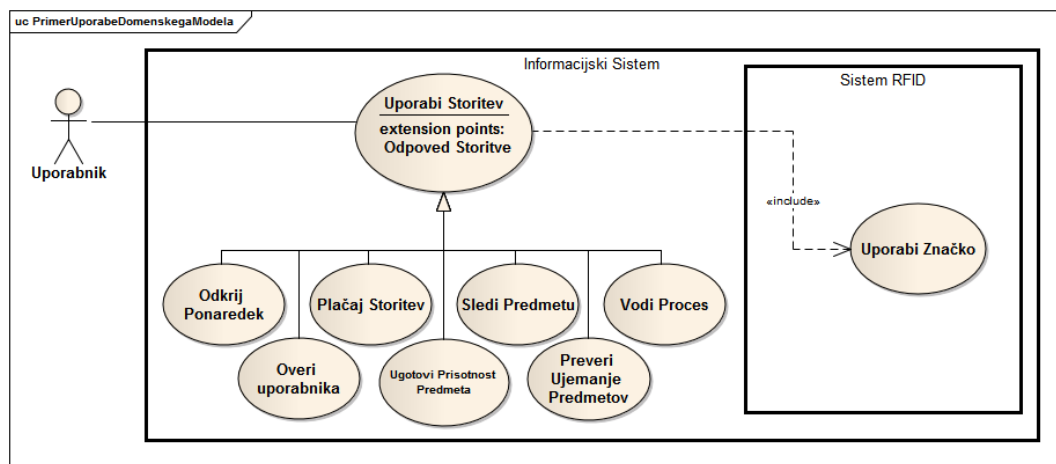
Dandanes so značke RFID že opremljene z različnimi senzorji, na primer za odčitavanje toplote. Podatek senzorike doda vrednost znački in se uporablja za različne poslovne in uporabniške namene. Današnja tehnologija omogoča povezovanje uporabnikov in predmetov z značkami kdajkoli in kjerkoli. Storitve globalnega imenika omogoča unikatno digitalno identiteto značk. V prihodnosti bodo značke sposobne medsebojnega povezovanja in zgodil se bo preskok v paradigmi povezovanja. Potekalo bo povezovanje med predmeti – kdajkoli in kjerkoli [10].

2.2 Opis tehnologije

Osnovni komponenti radiofrekvenčne identifikacije predstavljata značka in čitalec, ki se povezujeta prek radijskih valov. Prenosni kanal značke in čitalca sestavljajo oddajnik, sprejemnik in brezžični medij. Tipično čitalec izzove značko in prenos podatkov poteka z elektromagnetno interakcijo. Z elektromagnetno interakcijo poteka tudi prenos energije, če je značka pasivna in zahteva zunanji vir energije za vzbujanje. Čitalec se v nadaljevanju povezuje z zalednim delom sistema preko klasičnih komunikacijskih sistemov. Značka, čitalec in zaledni del sistema skupaj tvorijo informacijsko-komunikacijski sistem RFID oziroma na kratko sistem RFID.

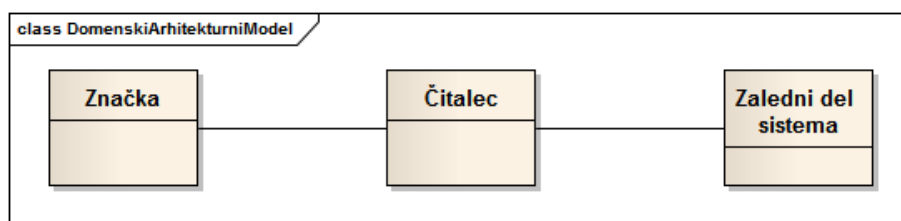
Medtem ko komunikacijski sistem povezuje komponente sistema RFID in navadno omogoča komuniciranje v smeri značke proti zalednemu delu sistema, se informacijski sistem uporablja za zajem, prenos, hranjenje, iskanje, obdelavo in prikazovanje podatkov. Končni uporabnik neposredno dostopa do storitev informacijskega sistema preko zalednega dela sistema. Informacijski sistem oskrbuje uporabnike z informacijami za odločanje in podporo delovnih procesov združbe na sledeči način. Identifikacijska številka predmeta se nahaja v znački predmeta in je zajeta s strani čitalca. Ta identifikacijski podatek je preoblikovan v obliko opisa, ki služi končnemu uporabniku in je prenešen v zaledni del informacijskega sistema, na primer z uporabo interneta. Podatkovna baza je osrednja

komponenta zalednega dela sistema, ki hrani opise dogodkov o zajetju za namen nadaljnega iskanja, obdelave in prikazovanja podatkov. Shranjeni dogodki o identificiranih predmetih se uporabijo za namen izvajanja delovnega procesa združbe (glej sliko 2.1).

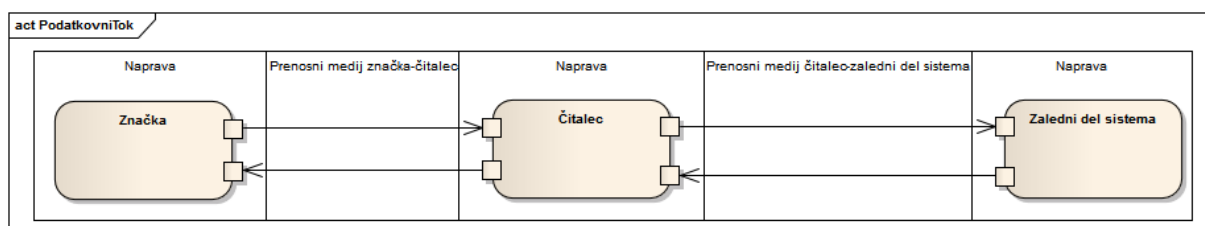


Slika 2.1: Diagram primera uporabe domenskega modela z razvrstitvijo legitimnih storitev.

Ta opis sistema RFID ujame bistvo funkcionalnosti (tehničnih) sistemov RFID, ki so podrobneje opisani v standardih ISO/IEC [11, 12, 13] in EPCglobal [14], a je obenem dovolj izčrpen, da služi kot podlaga za nadaljnjo analizo groženj in scenarijev napadov. Za ta namen sta ustvarjena naslednja modela: domenski arhitekturni model (glej sliko 2.2) in pripadajoči model podatkovnega toka s fizičnimi razmejitvami (glej sliko 2.3).



Slika 2.2: Razredni diagram domenskega arhitekturnega modela.



Slika 2.3: Diagram aktivnosti prikazuje tok podatkov domenskega arhitekturnega modela in pripadajoče fizične razmejitve.

2.3 Pomen kakovosti sistema

Prednosti uporabe sistemov RFID v poslovnih sistemih sta izboljšani učinkovitost in produktivnost določenih delovnih procesov na račun avtomatizacije in natančnosti pri identifikaciji predmetov. Prav tako je izboljšana učinkovitost upravljanja določenih delovnih procesov na podlagi realnočasnega spremljanja dogodkov o predmetih.

Navedene prednosti tehnologije RFID je mogoče uresničiti le ob zadostni kakovosti sistemov RFID. Po ISO 9000 [15] je to stopnja, na kateri množica karakteristik sistema izpolnjuje potrebe deležnikov. Le-ti določajo zahteve kakovosti sistema v uporabi, ki so izražene v stopnjah zmožnosti karakteristik sistema.

2.3.1 Karakteristike kakovosti

V zgornji razlagi vzrokov prodora tehnologije RFID je bil predstavljen primer učinka dveh tipov karakteristik sistema v uporabi, in sicer učinkovitost in produktivnost. Učinkovitost je zmožnost izdelka, ki deležnikom omogoča natančno in popolno doseganje zastavljenih ciljev v predpisanem kontekstu uporabe. Produktivnost je razmerje med doseženim učinkom in porabljenimi sredstvi deležnikov v predpisanem kontekstu uporabe. Sredstva deležnikov lahko predstavljajo vloženi čas in napor deležnikov, materialna sredstva ali finančni stroški. Model kakovosti ISO/IEC 9126-1:2001 [16] poleg teh dveh tipov določa še zadovoljstvo in varnost (*angl. safety*). Zadovoljstvo je zmožnost izdelka, da zadovolji uporabnike v predpisanem kontekstu uporabe. Varnost (*angl. safety*) je zmožnost izdelka, da zagotovi sprejemljivo stopnjo tveganja za poškodbe ljudi, združbe, programsko opremo, lastnino ali okolje v predpisanem kontekstu uporabe. Vzrok za nastanek tveganja so pomanjkljivosti v varnosti (*angl. security*), zanesljivosti, uporabnosti ali vzdrževanju izdelka.

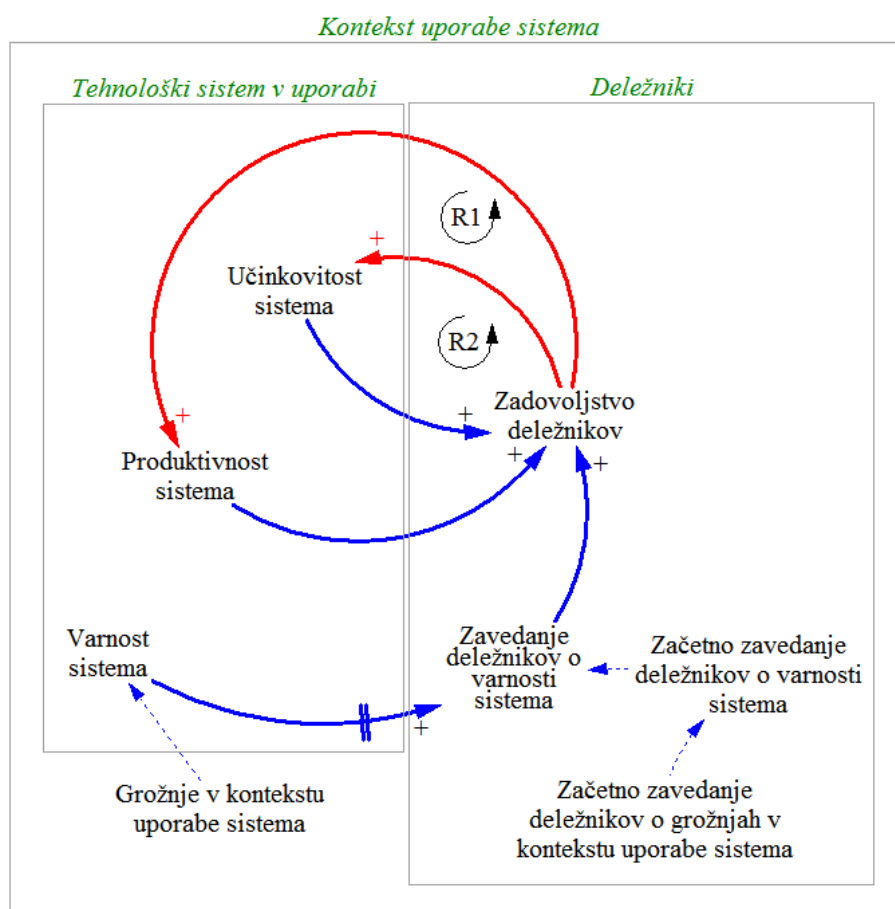
V okviru magistrske naloge se bomo osredotočili na en sam vzrok za nastanek tveganja, in sicer na pomanjkljivost v varnosti (angl. security) in na zagotavljanje varnosti (angl. security assurance) v procesu obvladovanja tveganja za zmanjšanje tveganja na sprejemljivo raven.

Še preden se dotaknemo zagotavljanja varnosti, je potrebno razumeti pomen varnosti z vidika uporabnika sistema. Zato si bomo v nadaljevanju ogledali vpliv varnosti na dinamiko obnašanja kakovosti sistema. Z drugimi besedami, zanimal nas bo vpliv varnosti na obnašanje ostalih karakteristik v daljšem časovnem obdobju.

Kadar bomo v nadaljevanju uporabili besedo sistem, bomo imeli v mislih socio-tehnični sistem, ki deležnikom omogoča uresničevanje lastnih potreb.

2.3.2 Odvisnosti karakteristik

Za razumevanje dinamike sistema je najprej potrebno spoznati medsebojno poveznost karakteristik. Učinkovitost, produktivnost in varnost skupaj vplivajo na zadovoljstvo deležnikov. Na diagramu 2.4 je to označeno z odebeljenimi modrimi puščicami. Vzročne povezave modre barve imajo naslednji pomen: večja kot je učinkovitost sistema, večje bo zadovoljstvo deležnikov. Medtem ko imata učinkovitost in produktivnost takojšnji učinek na zadovoljstvo deležnikov, ima po drugi strani varnost zakasneni učinek. Zakasnitev je na puščici označena z dvema vzporednima črtama. Pomeni, da zavedanje deležnikov o varnosti sistema kasni glede na poslabšanje varnosti sistema zaradi nastanka ranljivosti. Posledično kasnijo tudi zadovoljstvo deležnikov, učinkovitost in produktivnost.



Slika 2.4: Diagram vzroka in posledic karakteristik kakovosti sistema.

Deležniki si na samem začetku ustvarijo začetno zavedanje o grožnjah in varnosti sistema. To zavedanje temelji na preteklih izkušnjah, saj nimajo vpogleda v celoten kontekst delovanja sistema niti v specifično zgradbo sistema. To pripelje navadno do napačnih in preoptimističnih ocen tveganja deležnikov [17]. V primerjavi z realno varnostjo sistema ima zavedanje o varnosti sistema takojšen vpliv na zadovoljstvo deležnikov. Na primer, če se zavedanje varnosti spremeni zaradi varnostnega incidenta, to spremeni tudi zadovoljstvo deležnikov. Še več, zaradi dveh ojačujočih povratnih zank produktivnost-zadovoljstvo (R1) in učinkovitost-zadovoljstvo (R2) sledi spremembi zadovoljstva deležnikov domino

učinek, ki ga ponazarja naslednji hipotetčni scenarij.

Zaradi neprimerne rokojanja z napačno identificiranim objektom se zgodi nezgoda s smrtnim izidom. Sistem, ki je doslej deloval brezhibno in pravilno identificiral predmete, dejansko ne zagotavlja varnosti, kot pričakujejo deležniki. Deležniki uskladijo zavedanje o varnosti sistema in raven tveganja uporabe sistema postane nesprejemljiva. Posledično sledi padec zadovoljstva deležnikov. Zaradi slabega obvladovanja tveganja vodstva združbe se deležniki izognejo tveganju, tako da opustijo del avtomatiziranega delovnega postopka in ga zamenjajo z ročnim prepoznavanjem predmetov. Posledici te odločitve sta zmanjšanje učinkovitosti in produktivnosti sistema. Za enako delo je potrebno vložiti več napora ali celo več ne more biti opravljeno v celoti. Poslabšanje učinkovitosti in produktivnosti vpliva na povišanje cene storitve. Povpraševanje po storitvi se zmanjša zaradi nezadovoljstva odjemalcev. Nezadovoljstvo lastnikov vodi v zaključek scenarija, ki je prepuščen izbiri bralca.

Ne glede na izid scenarija pa sta za dinamiko obnašanja kakovosti sistema pomembni sledeči ugotovitvi:

- Zavedanje deležnikov o varnosti sistema in dejanska varnost sistema nista nujno enaki ves čas uporabe.
- Vpliv dejanske varnosti sistema na zadovoljstvo deležnikov je zakasnen in ima sistemski učinek.

2.4 Varnost

Varnost je zmožnost sistema, da doseže sprejemljivo raven tveganja za vse deležnike, tako da so zagotovljene njihove varnostne potrebe. Po standardu ISO/IEC 27000:2009 [18] je varnost zmožnost ohranjanja zaupnosti, celovitosti in razpoložljivosti informacij, poleg tega pa tudi ohranjanje drugih lastnosti, kot so verodostojnost, odgovornost, neovrgljivost in zanesljivost.

Vprašanja, ki si jih bomo v nadaljevanju zastavili, so povezana s posameznimi vidiki zagotavljanja varnosti.

- Kaj so vzroki za nastanek tveganja oziroma grožnje?
- Kako obravnavamo tveganja in katere protiukrepe imamo trenutno na voljo?
- Kako poteka proces obvladovanja tveganja?
- Kako dosežemo ustrezno stopnjo zagotavljanja varnosti?

V povezavi s predstavljenim modelom dinamike kakovosti sistema (glej sliko 2.4) nas bo zanimalo, kako učinkovito obvladovati tveganja in zagotavljati varnosti značk v sistemu

RFID in s tem izboljšati kakovost in varnost sistema. Na ta način bi se znižalo tveganje pri varovanju informacij, kar bi imelo pozitiven sistemski učinek ohranjanja zadovoljstva deležnikov, učinkovitosti in produktivnosti sistema.

2.5 Grožnje

Grožnje so možni vzrok nezaželenega incidenta, to je izkoriščanja različnih nepopolnosti. Grožnje so predpogoj za obstoj tveganja. Tveganje je po ISO/IEC 27000:2009 kombinacija verjetnosti nezaželenega incidenta in njegovih posledic. Sistem je varen, kadar doseže sprejemljivo raven tveganja za vse deležnike, tako da so zagotovljene njihove varnostne potrebe.

2.5.1 Od groženj do izkoriščanja ranljivosti

Za prepoznavanje tveganja je torej potrebno razumeti izvor grožnje, ki pretijo poslovnemu sistemu. Šolski primer napak pri obvladovanju tveganja opiše Anderson v članku "Why cryptosystems fail" [19] in se navezuje na primere izkoriščanja ranljivosti bančnih avtomatov v 70. in 80. letih prejšnjega stoletja. Načrtovalci sistema niso dobro določili konteksta uporabe sistema, zato so prepoznali le grožnje bankam. Groženj ostalim deležnikom pa niso prepoznali in so jih zanemarili. Največja grožnja bankam je bila ogrožitev informacij s prisluškovanjem in nepooblaščen uporaba opreme. V času hladne vojne so največji izvor groženj predstavljale tuje vladne agencije in velike združbe s sposobnostjo kriptanalize sporočil. Ta tveganja so bila zmanjšana s tehničnim protiukrepom za ohranjanje zaupnosti in celovitosti sporočil sistema, specifično s šifriranjem sporočil. Banke so se prepričale v ustrezno stopnjo zagotavljanja varnosti z uporabo kriptanalize. Banke so bile prepričane, da njihov sistem dosega sprejemljivo stopnjo tveganja, vendar so zaradi začetne napake pri obvladovanju tveganja prezrle druge grožnje, ki so se slej ko prej uresničile: socialni inženiring, tehnična okvara opreme, nepooblaščen uporaba opreme bančnih uslužbencev in napaka pri uporabi opreme bančnih uslužbencev. Čeprav izkoriščanje zadnjih ranljivosti ni povzročilo veliko škode s sistemskega vidika, so bili ti primeri vseeno medijsko zelo odmevni. Posledica je bila nezadovoljstvo deležnikov. Ta potek je skladen z modelom dinamike kakovosti sistema (glej sliko 2.4), saj se izkaže, da je bilo zavedanje deležnikov o varnosti sistema preoptimistično. Incidenti so se zgodili z določenim časovnim zamikom po postavitvi sistema. Nauk te zgodbe je, da je varnostne potrebe deležnikov potrebno prepoznati že pri načrtovanju in jih upoštevati pri razvoju in postavitvi sistema.

2.5.2 Grožnje omrežnih sistemov RFID

Zgodovina je slab učitelj varnosti. Trideset let po pojavu prvih bančnih avtomatov doživimo razcvet tehnologije RFID in ponovitev zgrešenih odločitev in napak obvladovanja

tveganja. “Elvis has left the border: ePassport faking guide unleashed” je le ena izmed odmevnih medijskih novic pri izkoriščanju ranljivosti sistema RFID [58]. Zaradi ranljivosti značke RFID je v 21. stoletju mogoče ponarediti potni list na doseg antene. Priključitev na internet in globalna dosegljivost značk interneta stvari pa bo napadalcem omogočala dodatne možnosti oddaljenega in avtomatiziranega napada [20].

Zgoraj opisani primer je realni prikaz napada na sistem RFID. Motivacija za napad kiberkriminalcev je zadovoljevanje njihovih potreb, ki imajo najrazličnejša ozadja, ki (poenostavljeno) segajo od terorističnih skupin pa vse do nezadovoljnih zaposlenih združbe. Cilj za doseganje zadovoljitve je izbrano sredstvo poslovnega sistema. Tehnologija pri tem navadno predstavlja le pot in ne končnega cilja za doseganje potreb. Iz tega sledi, da obstaja več možnih scenarijev napada na sistem RFID, ki imajo različne vplive na stanje varnosti poslovnega sistema.

Pri analizi scenarijev nas bodo zanimali neposredni napadi na značko oziroma njeno komunikacijo s čitalcem in vplivi napadov na poslovni sistem. Omejili se bomo na napade, ki so posledica človeškega dejavnika. Izbrani primeri napadov se nahajajo v spodnji tabeli (glej tabelo 2.1) in so razvrščeni glede na grožnje po tipologiji ISO/IEC 27005:2008. V zadnjem stolpcu tabele so navedene posledice posameznih napadov in vplivi na karakteristike varnosti in zasebnosti.

Poznavanje groženj je ključnega pomena za kasnejše obravnavanje tveganja in izbire zadostnih protiukrepov. Na podlagi zgornje tabele pridemo do sklepa, da grožnje znački in njeni komunikaciji s čitalcem vplivajo na stanje varnosti in zasebnosti celotnega poslovnega sistema. V splošnem lahko pričakujemo onipotentnega napadalca, ki ima fizični nadzor nad značko in sposobnost prisluškovanja komunikaciji s čitalcem ter možnost vnašanja in spreminjanja sporočil.

Tip grožnje	Grožnja	Primer napada	Učinek napada
Ogrozitev informacij	Prisluškovanje	Prisluškovanje komunikaciji značke in čitalca za nezakonito pridobivanje podatkov ciljnega trženja (angl. sniffing).	Zaupnost storitve sledenja predmetov, zasebnost imetnikov značke
Ogrozitev informacij	Kraja	Začasna odtujitev značke za pridobitev tajnega ključa (angl. physical side-channel attack), ponareditev značke in pretvarjanje (angl. spoofing) za pridobitev zaupne informacije.	Zaupnost in celovitost storitve nadzora dostopa
Ogrozitev informacij	Razkritje	Razkritje napake (angl. disclosure) algoritma za izračun tajnega sejnega ključa komunikacije značke in čitalca, ki se uporablja za sistem nadzora fizičnega dostopa vladne ustanove.	Celovitost storitve nadzora dostopa
Ogrozitev informacij	Nepooblaščen poseganje	Vpis podatka na značko (angl. insert), ki vsebuje zlonamerno kodo (angl. payload). Podatek se preko čitalca posreduje zalednemu delu sistema, kjer se izvede zlonamerna koda injekcija ukaza SQL.	Celovitost podatkovne baze sistema vodenja oskrbne verige
Ogrozitev informacij	Odkritje položaja	Sledenje položaju (angl. tracking) sovražnih bombnikov na podlagi ranljivosti značke RFID.	Zaupnost vojaške naloge in celovitost človeških življenj
Tehnična okvara	Nepooblaščen uporaba opreme	Namestitev virusa (angl. virus) na značko, zaradi katerega čitalec dobi vedno drugačen odziv elektronske kode produkta značke, ki vodi v odpoved storitve vodenja poslovnih procesov.	Celovitost storitve vodenja procesov
Nepooblaščen dejanje	Okvara podatkov	Vnos deformirane niza elektronske kode izdelka (angl. insert) za onemogočenje storitve sledenja predmetov poštnih pošilk.	Razpoložljivost storitve sledenja predmetov
Nepooblaščen dejanje	Nezakonita obdelava podatkov	Nezakonito spremljanje potovanja imetnikov kartice zaupanja v urbanem središču (angl. tracking) preko javne mreže čitalcev za namen ciljnega trženja.	Zasebnost posameznika
Nepooblaščen dejanje	Poneverba podatkov	Vnos veljavne elektronske kode produkta (angl. insert) za ponareditev izvirnosti modnega izdelka višjega cenovnega razreda (angl. counterfeiting).	Celovitost storitve preprečevanja ponaredkov
Ogrozitev funkcij	Poneverjanje pravic	Napadalec zasleduje lastnika avtomobila s ključem RFID. S pomočjo komunikacijskega releja poveča doseg komunikacije ključa – značke RFID in čitalca avtomobilske ključavnice (angl. relay). Njegov pajdaš ukrade avtomobil.	Celovitost storitve nadzora dostopa
Ogrozitev funkcij	Zavrnitev storitve	Uporaba motilnega signala (angl. jamming) za preprečevanje samodejnega plačevanja cestnin težkega tovornega prometa (angl. Denial Of Service – DoS)	Razpoložljivost samodejnega plačevanja cestnine

Tabela 2.1: Tabela groženj, napadov in učinkov po tipologiji namernih groženj ISO/IEC 27005:2008.

2.6 Protiukrepi

Potreba po izboljšanju stanja varnosti in zasebnosti na področju tehnologije RFID trenutno ne izhaja iz upoštevanja dobrih praks obvladovanja tveganj pri varovanju informacij kot notranji dejavnik, ampak iz medijsko odmevnih incidentov in nezadovoljstva deležnikov kot zunanji dejavnik, kar pričajo tudi številni primeri incidentov, povezanih z značkami [59, 60, 61, 58, 62, 63, 64, 65] in [21, 22].

Načrtovanje in razvoj protiukrepov sistemov RFID je zaradi heterogenosti in kompleksnosti uporabljenih tehnologij težavno v mnogih pogledih. Največjo težavo pri vpeljavi protiukrepov povzročajo omejitve značke, ki so računsko, pomnilniško, energijsko in komunikacijsko omejene naprave v primerjavi z ostalimi komponentami sistema. Dodatno, značke so specializirane strojne naprave, zato morajo biti protiukrepi načrtovani po meri. Zgornja cenovno dopustna meja značk je manjša od 10 evro centov [23], zato je maksimalno število logičnih vrat omejeno na nekaj tisoč. Od tega je približno 2000 vrat na voljo za načrtovanje in gradnjo protiukrepov. Medtem ko je predviden še nadaljnji padec cen značk, se zgornja dopustna meja logičnih vrat na značko zvišuje (Moorov zakon). Zaradi tega bo mogoče v prihodnosti uporabiti nove protiukrepe, ki jih bo potrebno preizkusiti preizkusiti, ali so pravilni in učinkoviti ter ali vnesejo novo ranljivost v sistem.

V literaturi so predlagani številni protiukrepi za izboljšanje stanja na področju varnosti in zasebnosti uporabe sistemov RFID. Uporaba močnih klasičnih kriptografskih gradnikov je stroškovno neugodna zaradi zgoraj navedenih omejitev značke. Zaradi tega so kriptografske rešitve optimizirane. Kompromisi lahko-kriptografskih rešitev se nahajajo v (i) produktivnosti izvajanja in (ii) v razporeditvi bremena računanja z značke na čitalec, kadar je le to mogoče. Lahka-kriptografija je aktivno področje znanstvenega raziskovanja in ponuja tudi (iii) nedeterministične rešitve kot kompromis med varnostjo in ceno protiukrepa. Pregled in opis možnosti uporabe nekaterih obstoječih lahko-kriptografskih gradnikov v protiukrepih značk se nahaja v [24].

Drugi predlogi protiukrepov ne vsebujejo kriptografskih gradnikov. Taka rešitev je na primer stikalo za uničenje značke in Faradayeva kleta (varovanje zasebnosti). Pregled izčrpne množice predlogov protiukrepov se nahaja v [25, 26, 27, 28, 29].

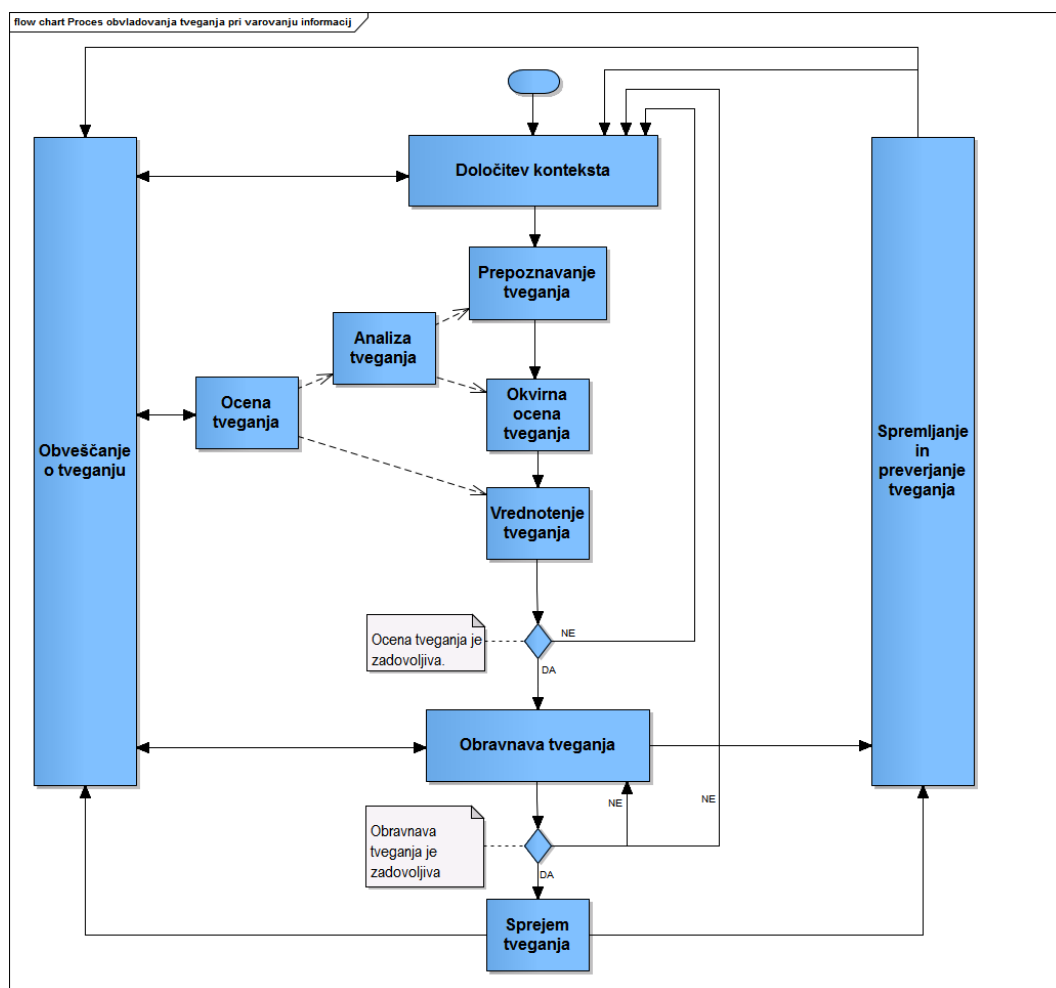
2.7 Obvladovanje tveganja

Proces obvladovanja tveganja pri varovanju informacij je po ISO/IEC 27005:2008 [30] sistematičen pristop za:

1. opredelitev potreb deležnikov glede varovanja informacij in
2. oblikovanje učinkovitega sistema za upravljanje varovanja informacij (SUVI) po postopku ISO/IEC 27003:2010 [31] za zagotovitev varnostnih zahtev ISO/IEC 27001:2005 [32].

V procesu obvladovanja tveganja (glej sliko 2.5) se tveganje prepozna, oceni, obravnava in sprejme, tako da je dosežena sprejemljiva raven tveganja deležnikov. Ocena tveganja je podana na podlagi meril za vrednotenje tveganja. Le-ta temelji na varnostnih potrebah deležnikov v okviru konteksta uporabe sistema. Tveganje se glede na oceno in merila obravnava na štiri načine.

- Raven tveganja se zmanjša z izborom protiukrepov.
- Tveganje se zadrži brez nadaljnjih ukrepov.
- Tvegani dejavnosti ali izvoru tveganja se izgone.
- Tveganje se prenese na zunanjo stranko, ki tveganje najbolj učinkovito upravlja.



Slika 2.5: Proces obvladovanja tveganja za varovanje informacij ISO/IEC 27005:2008.

Rezultat te aktivnosti je načrt za obravnavano tveganja. Sprejetje tega načrta pomeni določitev odgovornosti za vzpostavitev učinkovitih in upravičenih protiukrepov, ki zmanjšajo tveganje na sprejemljivo raven in zadovoljijo varnostne zahteve. Vse dejavnike tveganja je potrebno spremljati in pregledovati, da se spremembe v sistemu zgodaj prepoznajo in

da se vzdržuje celostni vpogled v tveganje. V primeru sprememb je potrebno ukrepati, tako da so sprožene določene aktivnosti procesa obvladovanja tveganja. Odgovorni za obvladovanje tveganja morajo izmenjati vse informacije v zvezi z dejavnostjo obvladovanja tveganja, saj je obveščenost podlaga za sprejem pravih odločitev. Na ta način se doseže sprejemljivo raven tveganja za vse deležnike, tako da so zagotovljene vse varnostne potrebe.

Protiukrepi so sredstva za upravljanje tveganja in se po tipologiji ISO/IEC 27002:2005 [33] uvrščajo med politike, postopke, smernice, prakse ali organizacijske strukture, ki so lahko upravne, tehnične, vodstvene ali pravne narave. V ISO/IEC 27002:2005 so navedeni primeri protiukrepov skupaj z njihovimi cilji učinkov na tveganje ter priporočili za vpeljavo v sistem.

V okviru magistrske naloge bomo v prvi vrsti osredotočeni na obvladovanje tveganja pri varovanju informacij proti namernim grožnjam. Osredotočeni bomo na upravljanje tveganja s protiukrepi, ki so tehnične narave.

Protiukrepi so idealno vpeljeni v sistem, tako da so varnostne potrebe izpolnjene v vsakem trenutku. Realno pa je te potrebe v procesu obvladovanja tveganja težko izpolniti zaradi prisotnosti človeške napake. Zaradi teh napak deležniki ne smejo slepo verjeti združbam ali razvijalcem, da so njihove varnostne zahteve zagotovljene.

2.8 Zagotavljanje varnosti

Rezultat procesa zagotavljanja varnosti je argument, ki predstavlja osnovo za zaupanje deležnikov, da izdelek, proces ali storitev izpolnjuje njihove varnostne potrebe. Namen tega procesa je pridobitev zaupanja deležnikov v varnost sistema in s tem v učinkovitost sistema za upravljanje varovanja informacij. Deležniki morajo določiti stopnjo zagotavljanja varnosti izdelka, procesa ali storitev, tako da opredelijo strogost in obseg ocenjevanja varnosti. Trditve o varnosti morajo biti podvržene evalvaciji, ki upošteva zahteve ocenjevanja. Argument je rezultat evalvacije varnosti.

Metode zagotavljanja varnosti se med seboj ločijo po strogosti in obsegu evalvacije. Uspešnost metode je pri tem odvisna od vsakega primera evalvacije posebej. Ne le zaradi specifičnih tehničnih lastnosti sistemov, temveč tudi zaradi različnih kontekstov uporabe in s tem povezanih različnih groženj [34].

Varnosti sistema ni mogoče zagotoviti z izoliranim preizkušanjem posamezne komponente, saj se posledice napak v načrtovanju, razvoju in postavitvi posamezne komponente odražajo kot odpoved varovanja informacij celotnega sistema [66].

Ob predpostavki, da so varnostne zahteve glede na kontekst uporabe pravilno določene in izbrani ustrezni protiukrepi nas bo v magistrski nalogi zanimalo zagotavljanje varnosti za okolja omrežnih sistemov RFID z verifikacijo in validacijo protiukrepov značk RFID.

V nadaljevanju bomo obravnavali (i) različne pristope za zagotavljanje varnosti, (ii) splošne zahteve orodij za preizkušanje varnosti in (iii) uveljavljene smernice načrtovanja aktivnosti preizkušanja varnosti. Prepoznane splošne zahteve bodo postavile okvir za nadaljnji razvoj in zasnovo magistrske naloge.

2.8.1 Tipologija metod zagotavljanja varnosti

Obstajajo različni tipi metod zagotavljanja varnosti, vsak s svojimi prednostmi in slabostmi. Ročna varnostna verifikacija je učinkovita, vendar ni izčrpna in je draga. Formalne metode so učinkovite za izčrpno avtomatizirano verifikacijo manjših komponent sistema in modela sistema. Zlasti pri formalni verifikaciji modela je težko zagotoviti, da bo obnašanje modela enako končni implementaciji. Metode avtomatizirane statične analize so izčrpne, vendar jih je v porazdeljenih heterogenih sistemih mogoče uporabiti le za verifikacijo posameznih komponent. Zadnji tip avtomatizirane verifikacije je testiranje delujočega sistema. Težavo pri tem predstavlja heterogena sestava sistema, pomanjkanje orodij za izvedbo testiranja in posebnost tehnologije RFID – testnih primerkov značk v razvoju dolgo ni na voljo za testiranje. Za testiranje je namreč potrebno imeti delujoči prototip značke ali pa le-to simulirati v realnem času [35, 36, 37]. Pri simulaciji je težko zagotoviti, da bo obnašanje programskega modela enako končni implementaciji sistema. Po drugi strani je gradnja prototipa zamudna, draga in podaljša čas prihoda na trg. Kadar deležniki ocenijo, da je tveganje izgube profita na račun poznega prihoda na trg večja od tveganja posledic varnostnega incidenta, potem uporaba prototipa za zagotavljanje varnosti ni upravičena. Neprimerno ali slabo testiran produkt navadno ne dosega sprejemljive ravni tveganja za deležnike.

2.8.2 Zahteve orodij za preizkušanje varnosti

Orodja za testiranje varnosti morajo biti dovolj prilagodljiva, da je mogoče izzvati obnašanje sistema, ki ni navedeno v naročnikovih zahtevah. Na ta način se odkrijejo ranljivosti sistema. Druge vrste funkcionalnega testiranja so manj učinkovite pri odkrivanju ranljivosti, saj so testni primeri sestavljeni, tako da preverjajo prisotnost obnašanja, ki je navedeno v naročnikovih zahtevah [38]. Prav tako pomembno je testiranje varnosti celotnega sistema, saj v nasprotnem primeru, s testiranjem posameznih komponent, ostanejo določene sistemske ranljivosti prikrite [66].

2.8.3 Načrtovanje aktivnosti preizkušanja varnosti

Skupnost strokovnjakov za informacijsko varnost je enotna, da se je problema obvladovanja tveganja potrebno lotiti na samem začetku načrtovanja sistemov ali produktov [33]. Tudi stroški obravnave tveganja in s tem povezanih sprememb so na začetku načrtovanja sistema nižji. V sledečih fazah življenjskega cikla produkta stroški obravnave ranljivosti eksponentno naraščajo [39]. To velja še posebej za sisteme RFID, v katerih značke

ni mogoče posodobiti ali zamenjati na enostavn način, ko se enkrat nahaja v operativni uporabi. V [40] avtorja argumentirata pozitivne učinke zgodnje vpletenosti procesa obvladovanja tveganja in zagotavljanja varnosti v razvoju produkta. Tako se na samem začetku prepoznajo primeri zlorabe in opravi se analiza poslovnih tveganj. Le-ta je vodilo za analizo tveganja načrtovane arhitekture. Po obravnavi tveganja poteka funkcionalno varnostno testiranje protiukrepov. Če se testi zaključijo pozitivno, je naslednji korak testiranje pravilnosti in učinkovitosti protiukrepov za upravljanje tveganja sistema. Testni primeri so vzpostavljeni na podlagi prepoznanih tveganj in oponašajo pametnega napadalca. Po namestitvi sistema so nazadnje opravljeni penetracijski testi. Vizija prihodnosti varovanja sistemov RFID je skladna z zgodnjo vpletenostjo procesa obvladovanja tveganja in zagotavljanja varnosti v razvoju sistema [26]. Primer tako izvedenega testiranja varnosti je bil opravljen na tehnologiji Java Card [41]. Obstajajo tudi druge pobude, kjer bi se objavili natečaji za nagrajevanje odkritih ranljivosti sistemov RFID [42], podobno kot v primeru spletnih brskalnikov, vendar ni nizkocenovnega orodja, ki bi strokovnjakom omogočal preizkušanje varnosti.

2.9 Cilj magistrske naloge

Cilj magistrske naloge je zasnova in razvoj orodja in metode za zagotavljanje varnosti za okolja omrežnih sistemov RFID. Orodje in metoda morata upoštevati trenutne in bodoče grožnje, ki bi izkoristile ranljivosti značk RFID. Zagotavljanje varnosti naj poteka v okviru obvladovanja tveganja s preizkušanjem varnosti na podlagi prepoznanih tveganj sistema s simulacijo napada na sistem. Simulacija napada naj omogoča testiranje funkcionalnosti varnostnih protiukrepov, testiranje varnosti na podlagi prepoznanih tveganj in penetracijsko testiranje sistema. Na ta način preizkuševalec prepozna ranljivosti in preizkusi pravilnost in učinkovitost protiukrepov.

METODA IN ORODJE ZA ZAGOTAVLJANJE VARNOSTI omrežnih sistemov RFID morata upoštevati trenutne in bodoče grožnje, ki bi izkoristile ranljivosti značk RFID. Izkoriščanje teh ranljivosti vpliva na varovanje informacij in zasebnost sistema ter posledično na stanje tveganja poslovnega sistema.

Zasnova in razvoj orodja morata upoštevati zgradbo in delovanje sistema RFID, prepoznane grožnje in napade. Preizkušanje varnosti naj poteka z orodjem po metodi za zagotavljanje varnosti v nadzorovanem okolju. Preizkušanje varnosti je simulacija napada na sistem na podlagi prepoznanih sredstev in groženj. Moč napada je omejena in sorazmerna prepoznanim grožnjam. Preizkušanje varnosti se uporabi v sklopu testiranja funkcionalnosti varnostnih protiukrepov ali penetracijskega testiranja. Na podlagi rezultata preizkušanja so določene ranljivosti sistema in podana je ocena tveganja sistema. Glede na oceno tveganja se izvedejo nadaljnji koraki obvladovanja tveganja pri varovanju informacij.

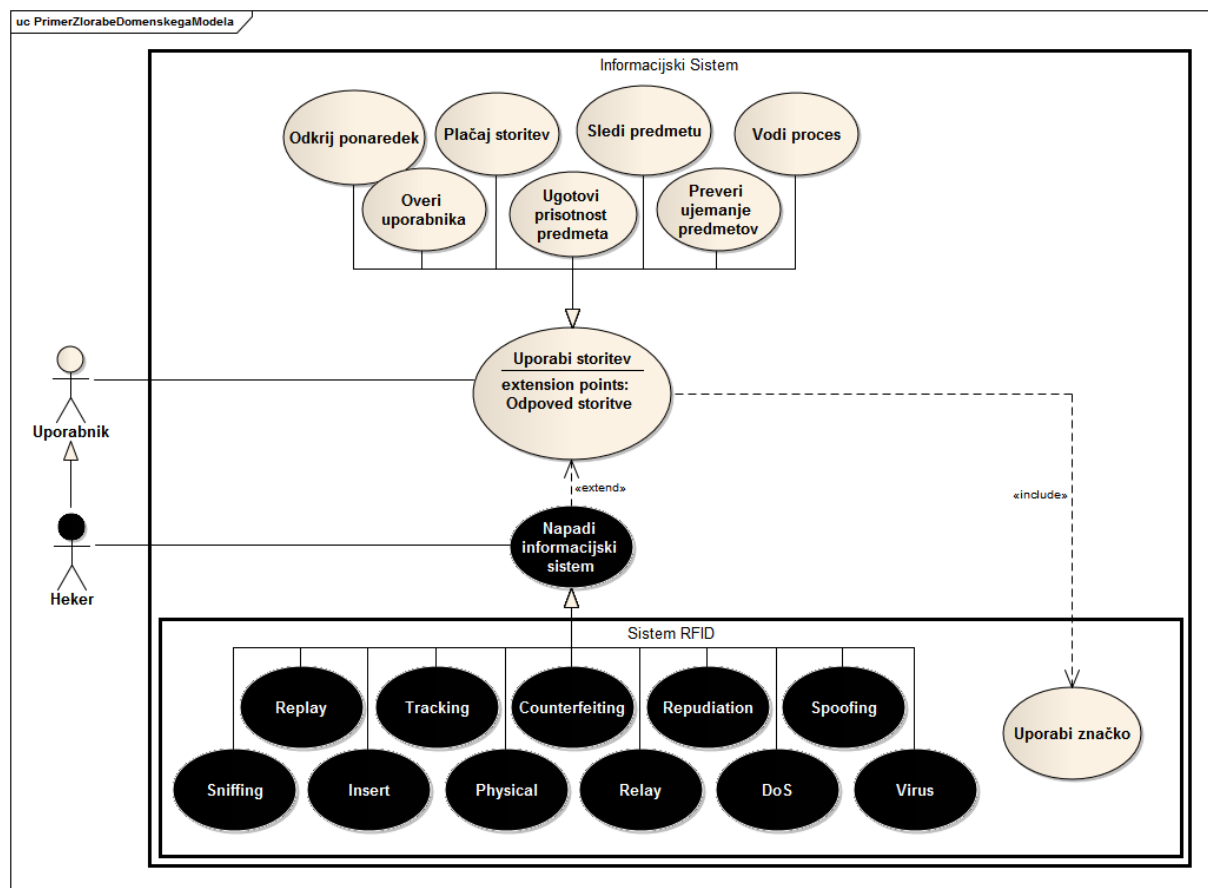
V tem poglavju bo najprej predstavljena analiza in specifikacija zahtev za zasnovo orodja. Izpolnitev teh zahtev bo glavni predmet obravnave tega poglavja. Opisana bo arhitektura orodja, v kateri bodo predstavljene posamezne komponente z vmesniki in njihova medsebojna interakcija. Podan bo potek preizkušanja varnosti sistema na demonstracijskem sistemu. Čisto na koncu poglavja bo definirana metoda zagotavljanja varnosti, ki bo umestila uporabo orodja v proces obvladovanja tveganja pri varovanju informacij.

3.1 Analiza zahtev

Najprej bomo pregledali funkcionalne zahteve orodja, ki bodo dale okvir zahtev. Pri realizaciji okvira bomo naleteli na številne tehnične omejitve in sprejeli bomo načrtovalske odločitve, ki bodo vplivale na končni načrt orodja. V sklopu omejitev bomo obravnavali: (i) omejitve pri realizaciji značke, (ii) omejitve aparaturnega priklopa značke na medij, (iii) omejitve pri realizaciji delovnega mesta preizkuševalca in (iv) optimizacijo poteka preizkušanja.

3.1.1 Funkcionalne zahteve

Orodje za preizkušanje varnosti predstavlja infrastrukturno podporo za simulacijo napadov hekerjev. Napadi so zbrani na sliki 3.1 in so povzeti po tabeli groženj, napadov in učinkov 2.1. Napadi predstavljajo razširitev uporabe sistema in so na sliki predstavljeni kot črno označeni primeri uporabe. Poimenovanje napadov je ohranjeno v angleškem jeziku zaradi prepoznavnosti terminov.



Slika 3.1: Diagram primera uporabe z razvrstitvijo napadov in legitimnih storitev.

Vse napade je mogoče izvesti in s tem zadostiti funkcionalnim zahtevam orodja, če orodje za preizkušanje varnosti omogoča preizkuševalcu popolni nadzor nad:

1. komunikacijskim kanalom značke in čitalca,
2. izvajanjem logičnih operacij značke.

3.1.2 Omejitve pri realizaciji značke

Naslednji dve oviri preprečujeta dostopnost uporabe orodja širšemu krogu raziskovalcev.

- Izdelava prototipa značke z vezjem ASIC (vezje za določeno aplikacijo) predstavlja strošek in zakasnitev pri razvoju značke.
- Generično orodje, ki bi omogočalo preizkušanje varnosti sistema z neposredno uporabo vezij ASIC, bi moralo vsebovati množico različnih aparaturnih priklopov na medij za komunikacijo z značko. Ti priklopi bi morali biti skladni z aparaturnim priklopom preizkušane značke. Izdelava takšnega orodja bi bila stroškovno neučinkovita.

Rešitev obeh težav je uporaba (i) realnočasne simulacije logike značke ali (ii) emulacija značke z vezjem FPGA. Še preden bomo primerjali obe možnosti za namen preizkušanja, bomo za boljše razumevanje povzeli zgradbo in delovanje vezja FPGA.

Vezje FPGA [67] je sestavljeno iz (i) logičnih celic, (ii) povezovalne matrike in (iii) vhodno-izhodnih blokov. Logična celica vsebuje vpogledno tabelo in pomnilno celico. Vpogledna tabela je programibilna in realizira poljubno n-vhodno logično funkcijo. Pomnilna celica realizira D celico in je nastavljiva za (i) nivojsko proženje (zapah) ali (ii) robno proženje (flip-flop). Vezje FPGA lahko vsebuje tudi druge funkcijske elemente, kot so na primer pomnilnik RAM, možilnik, integrirani procesor. Povezovalna matrika je programirljiva in služi za povezovanje funkcijskih elementov [43].

V [44, 45] se nahaja pregled prednosti in slabosti izbire simulacije ali realizacije v FPGA za verifikacijo digitalnih vezij in priporočena industrijska praksa. Te prednosti in slabosti bomo uporabili v navezavi z orodjem za preizkušanje varnosti.

Prednost simulacije je programska dostopnost stanja logike. Časovna ločljivost izvajanja simulacije je ena urina perioda značke. Nadzor nad logičnim izvajanjem značke in komunikacijo značke s čitalcem poteka z branjem in pisanjem spremenljivk programa. Razpoložljivost programskih knjižnic za razhroščevanje je dodatna prednost uporabe simulacije. Simulacija je počasna za večja digitalna vezja, vendar to ni ovira za simulacijo značk, saj je zgornja meja števila logičnih vrat zaradi stroškov izdelave zelo omejena. Večjo oviro predstavljajo lastniški simulatorji [68, 69, 70, 71, 72, 73] z zaprtim programskim vmesnikom API, ki v večini ne podpirajo realnočasne simulacije, odprtokodne rešitve [74] pa se zaradi realnočasnih zahtev izkažejo za nezadostne.

Izbrana alternativa je emulacija značke v FPGA. Slabost emulacije je dostopnost logike in signalov značke, ki so sedaj strojna oprema. Združbi Xilinx in Altera sta za ta namen razvili logični analizator za vezja FPGA [75, 76], vendar imata obe rešitvi zaprti aplikacijski programski vmesnik API, kar onemogoča nadaljnje povezovanje z orodjem in s čitalcem ter onemogoča izpolnitev ene od zahtev orodja. Problem povezovanja značke bo potrebno rešiti na drugačen način, s strojno in programsko opremo po meri.

Za opis delovanja značke je izbran strojno opisni jezik VHDL [77]. Ta jezik služi za opis strukture in obnašanja digitalnih sistemov. Na podlagi opisa VHDL je mogoče vezja simulirati. Na podlagi opisa VHDL je prav tako mogoče vezja realizirati v FPGA in ASIC. Še več, obstaja lastnost ekvivalence preslikave tehnologije med vezji FPGA in vezji ASIC [46].

Učinkovitost orodja za preizkušanje varnosti bo odvisna od izvedbe emulacijskega sistema in razpoložljivosti opisa značke v strojnem opisnem jeziku.

3.1.3 Omejitve aparaturnega priklopa značke

Emulacija logike značke ne reši problema aparaturnega priklopa na medij. Še vedno bi potrebovali množico različnih aparaturnih priklopov, kar bi bilo stroškovno neugodno in bi otežilo realizacijo delovnega mesta preizkuševalca. Rešitev aparaturnega priklopa značke je naslednja. V sklopu referenčnega modela ISO OSI se fizična plast značke nadomesti s fizično plastjo emulacijskega okolja. Fizična plast skrbi za prenos bitov preko prenosnega medija in zagotavlja standardno aparaturno priključevanje značke na prenosni medij.

V prvem primeru je medij prazen prostor, aparaturni priključek pa je naprava radio. V drugem primeru je medij električni vodnik (metalizacija integriranega vezja), aparaturni priključek pa je izravnalnik napetostnih nivojev (angl. buffer). Novi komunikacijski kanal, ki ga sestavljata dva izravnalnika in metalizacija integriranega vezja, bo imel naslednje lastnosti:

- sočasno dvosmerni kanal,
- kanal ne bo dopuščal kolizij,
- serijski kanal,
- digitalni kanal,
- kanal bo omogočal priključitev dveh aktivnih elementov na en segment prenosnega medija in
- sinhronizacijo pri prenosu vsakega znaka.

Emulacijsko okolje bo zagotavljalo povezovanje med značko in čitalcem s tuneliranjem. To tuneliranje bo transparentno za značko in čitalec. Fizično plast je potrebno nadomestiti tudi v čitalcu. Ker bo transparentno tuneliranje služilo za povezovanje med značko in čitalcem, bo zahtevana sprememba čitalca programska. Potrebno bo spremeniti proceduro za pisanje in branje iz komunikacijske naprave čitalca.

3.1.4 Omejitve pri realizaciji delovnega mesta preizkuševalca

Strojna oprema emulacijskega okolja je razvojno okolje Xilinx XUP Virtex-II Pro [78]. Razvojno okolje je dovolj zmogljivo, da podpira emulacijsko okolje, a premalo, da bi podpiralo delovno mesto preizkuševalca in realiziralo interaktivni grafični uporabniški vmesnik. Iz tega razloga je potrebno vire orodja razpršiti. Pri tem je aplikacija odjemalca orodja realizirana na osebem računalniku, aplikacija strežnika pa na razvojnem okolju FPGA.

Razvojno okolje Xilinx XUP Virtex-II Pro ponuja tri možnosti povezovanja porazdeljenega orodja.

- Tesno povezovanje virov poteka z naslovnim, podatkovnim in kontrolnim vodilom (na primer protokol in vodilo PCI).
- Ohlapno povezovanje virov poteka s podatkovnim in kontrolnim vodilom (na primer protokol UART z RS-232 ali protokol JTAG).
- Omrežno povezovanje virov poteka s podatkovnim vodilom (na primer Ethernet IEEE 802.3i ali Ethernet IEEE 802.3u).

Razvojno okolje ne vsebuje aparaturnega priključka PCI, zato tesno povezovanje ni stroškovno učinkovito. Realizacija ohlapnega povezovanja je zaradi preproste izvedbe protokola enostavna, vendar ni razširljiva. Po drugi strani je realizacija omrežnega povezovanja težja, saj zahteva razpoložljivost osnovnega nabora internetnih protokolov. Vendar ima prednost, saj je ta rešitev potem razširljiva.

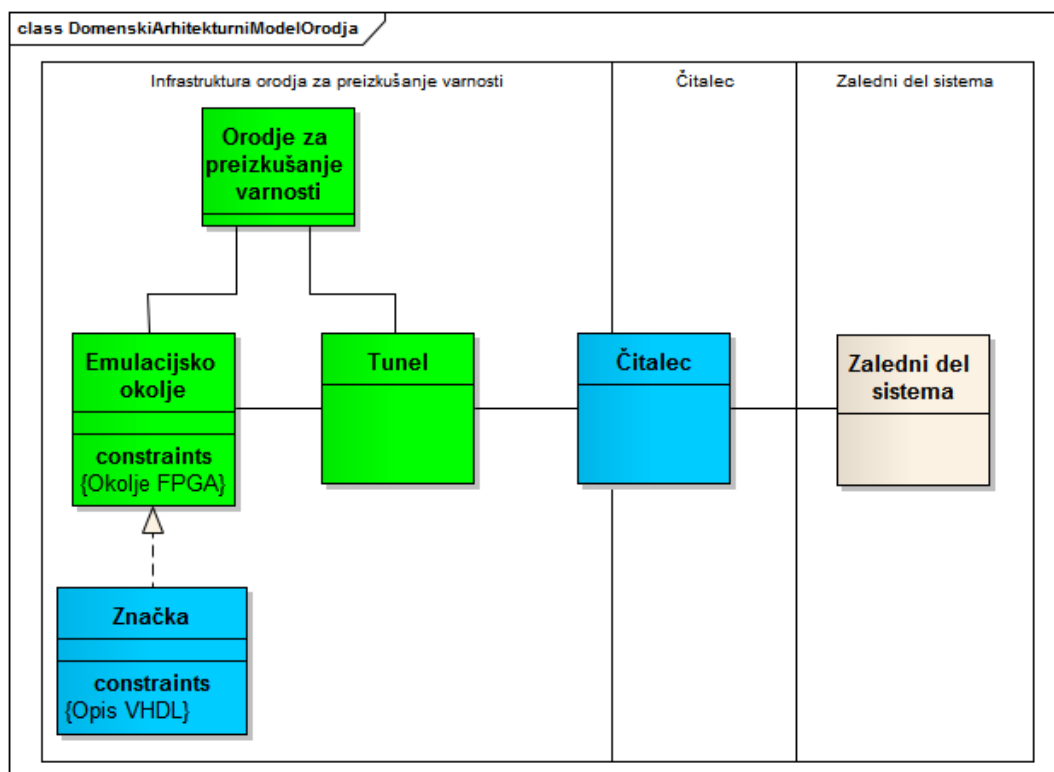
Zaradi boljše razširljivosti in mobilnosti orodja smo izbrali omrežno povezovanje virov. Za realizacijo aplikacije strežnika bo uporabljen vgrajeni sistem, ki ga bo sestavljal vgrajeni procesor in operacijski sistem s podporo omrežnega povezovanja.

Zgoraj opisana aplikacija strežnika bo sestavni del emulacijskega okolja, ki se bo povezovalo z orodjem za zagotavljanje varnosti (glej sliko 3.2). Skladno s funkcionalnimi zahtevami bo orodje omogočalo nadzor nad (i) komunikacijskim kanalom značke in čitalca in (ii) logičnim izvajanjem značke.

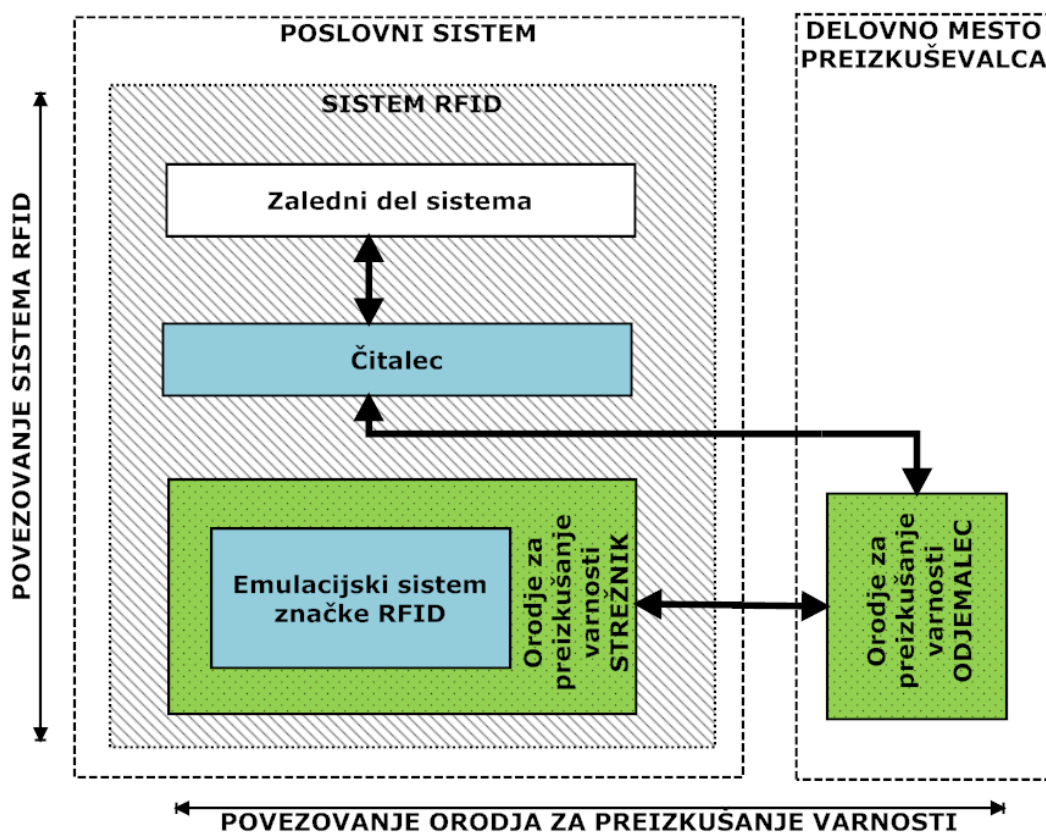
3.1.5 Optimizacija poteka preizkušanja

Integracija orodja s sistemom je mogoča ob razpoložljivem opisu značke v strojno opisnem jeziku in ob zgoraj opisanih transformacijah značke in čitalca. Na sliki 3.2 sta (delni) spremembi vozlišč označeni z modro barvo. Dodana vozlišča so označena z zeleno barvo. To so orodje za preizkušanje varnosti, emulacijsko okolje in komunikacijski tunel. Zaledni del sistema se ne spremeni.

Po tem, ko je postopek integracije orodja s sistemom opravljen, preizkuševalec na delovnem mestu odkriva ranljivosti poslovnega sistema, tako da izvaja napad nad (i) komunikacijskim kanalom značke in čitalca ali (ii) nadzira logično izvajanje značke. Orodje je najbolj učinkovito, če je oba napada mogoče koordinirati, zato je namesto dveh ločenih vmesnikov (glej sliko 3.2) smiselno uporabiti enoten vmesnik na izvajanje nadzora nad značko. Podatkovni tok takšnega sistema je prikazan na sliki 3.3.



Slika 3.2: Domenski arhitekturni model orodja.



Slika 3.3: Bločni diagram načina preizkušanja z emulacijo.

3.2 Arhitektura orodja

Na podlagi specifikacije zahtev je ustvarjen arhitekturni model orodja. V opisu arhitekture bodo opredeljene (i) komponente, (ii) njihove funkcionalnosti, (iii) načrt komponente, (iv) njihova hierarhija, (v) njihovi vmesniki in (vi) njihovo logično povezovanje z drugimi komponentami. Arhitekturni model orodja je prikazan na diagramu postavitve 3.4.

Opis komponent bo potekal v smeri podatkovnega toka od značke do zalednega dela sistema. Najprej bo podan splošni opis komponente, nato pa opis realizacije demonstracijskega sistema.

3.2.1 Kontekst demonstracijskega sistema

Kontekst demonstracijskega sistema je združba obrambne industrije, ki izdeluje in opremlja vojsko z vojaško opremo. Združba uporablja sistem za kontrolo pristopa v prostore podjetja, ki je realiziran z omrežnim sistemom RFID.

Vsak zaposleni dobi brezžično značko, ki vsebuje elektronsko kodo produkta. Zaposleni pristopi k prehodu in približa značko čitalcu. Čitalec posreduje elektronsko kodo produkta in identifikacijsko številko čitalca zalednemu delu sistema. Ta overi in avtorizira uporabnika, tj. preveri identiteto nosilca značke in preveri, ali ima zaposleni dodeljeno vlogo, ki omogoča izvedbo operacije, npr. odklep vrat.

3.2.2 Emulacijsko okolje

Osrednji element emulacijskega okolja predstavlja značka RFID. Opis značke se nahaja v jeziku (i) VHDL, (ii) Verilog ali (iii) SystemC in opisuje zgradbo ter delovanje vezja na ravni abstrakcije RTL. Ta abstrakcija vezja se uporablja za opisovanje sinhronnega digitalnega vezja, tako da obnašanje vezja določajo podatkovni toki med sinhronimi robno proženimi pomnilnimi celicami. Nad podatkovnimi tokovi se izvajajo logične operacije.

Slika 3.5 prikazuje generično strukturo vezja RFID. Osnovni opis vezja RFID je potrebno dopolniti s krmilnimi elementi emulacijskega okolja, ki so na sliki označeni z zeleno prekinjeno črto. Le na ta način je omogočen nadzor nad logičnim izvajanjem značke in njeno komunikacijo s čitalcem.

Dopolnjeni opis značke je pripravljen za vnos v emulacijsko okolje, ki je prikazan na sliki 3.6. Emulacijsko okolje sestavljajo (i) značka RFID, (ii) krmilnik emulacijskega okolja in (iii) adapter za povezovanje z emulacijskim okoljem IPIF [79].

Adapter IPIF je vezni člen med vodilom procesorja MicroBlaze in emulacijskim okoljem. Adapter zagotavlja aparaturni priklop na vodilo in vsebuje dva izravnavalnika za boljšo produktivnost orodja. Glede na smer uporabniškega povezovanja z emulacijskim okoljem sta izravnavalnika poimenovana pisalni FIFO (na sliki WFIFO) in bralni FIFO (na sliki

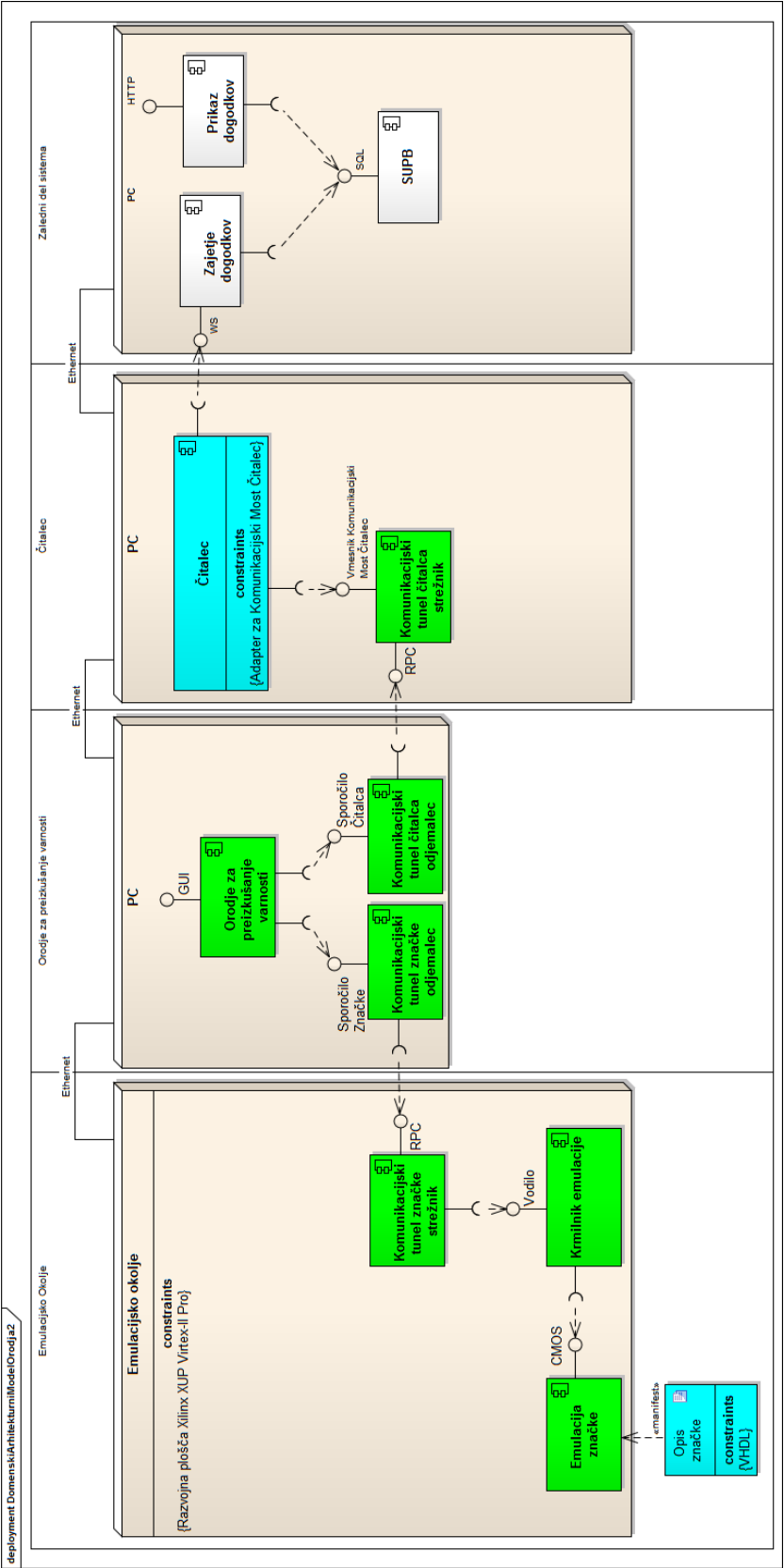
RFIFO). Izravnalnika sta realizirana s krožno vrsto FIFO, ki ima končno kapaciteto. Zaradi zahtevane zanesljivosti orodja sta vrsti realizirani, tako da vpis podatka v polno vrsto ne povzroči prelitja. Krožna vrsta FIFO prav tako reši sinhronizacijski problem proizvajalec-porabnik, to je problem sočasnega pisanja in branja dveh paralelnih procesov.

Krmilnik emulacijskega okolja je realiziran z determinističnim končnim avtomatom. V začetnem stanju avtomat prevzame ukaz iz vrste WFIFO in ga dekodira (korak 1 in 2). V naslednjem stanju avtomat izvrši ukaz, tako da izvede eno od dveh osnovnih operacij in omogoči uro značke za obdobje ene urine periode (korak 3).

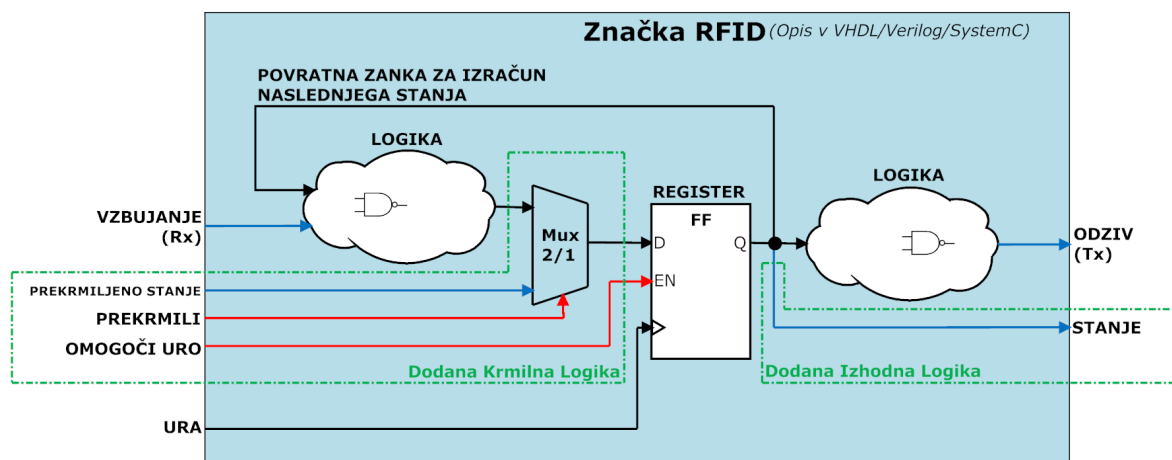
- Naslednje stanje značke je odvisno od trenutnega stanja značke in vrednosti vhodov značke. Naslednje stanje določa rezultat logične funkcije za izračun naslednjega stanja f_{RFID} . Izhod značke določa rezultat logične funkcije g_{RFID} .
- Naslednje stanje značke je prekrmljeno in določeno z vrednostjo takojšnjega operanda, ki se nahaja v ukazu. Izhod značke določa rezultat logične funkcije g_{RFID} .

Avtomat v zadnjem koraku onemogoči uro značke in zapiše rezultat ukaza v vrsto RFIFO (korak 4), kjer se podatki nahajajo do prevzema.

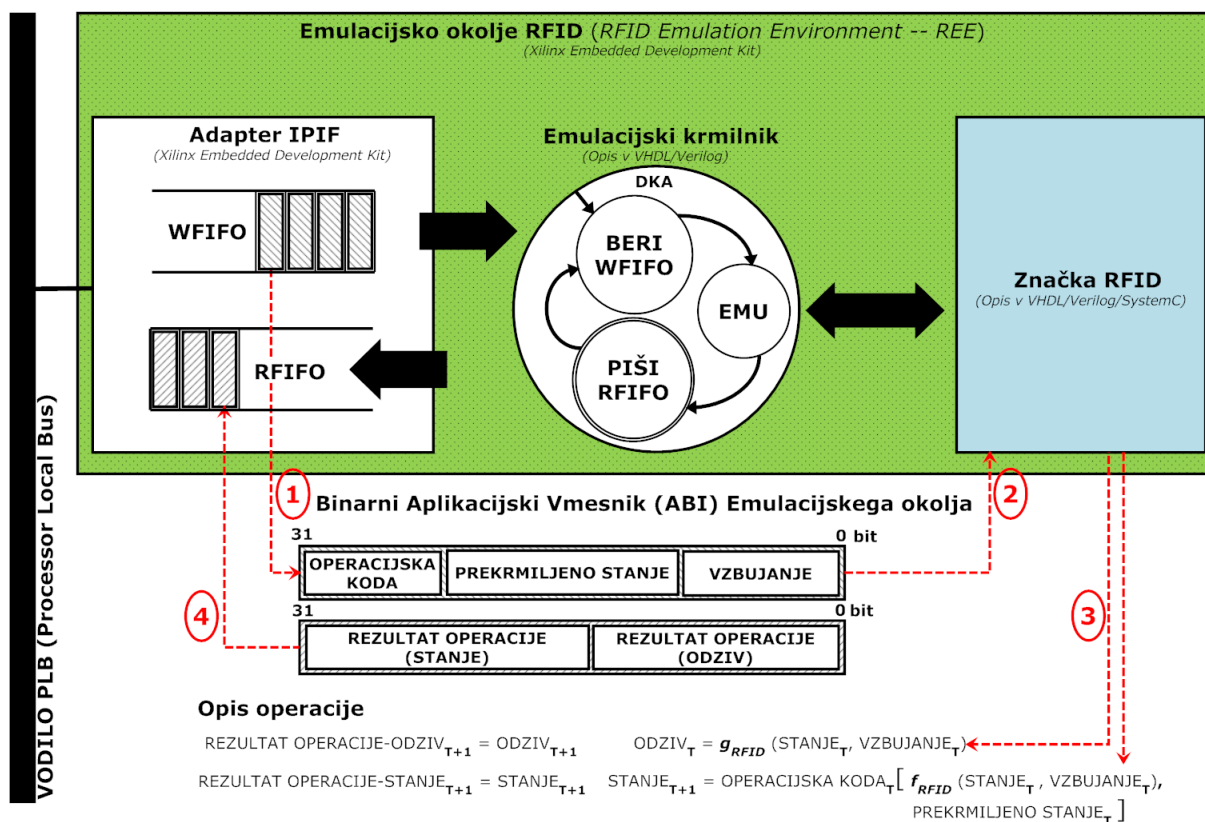
Ukaz ima fiksno dolžino in je omejen z 32-bitno širino vodila, na katero je priključeno emulacijsko okolje. Operacijska koda se nahaja na mestu najbolj pomembnega bita ukaza. Operandi so eksplicitni in takojšnji. Njihova število in dolžina sta v splošnem prilagodljiva. Ponor rezultata operacije je določen implicitno in se vedno zapiše v izhodno vrsto RFIFO.



Slika 3.4: Arhitekturni model orodja.



Slika 3.5: Generična struktura vezja RFID.



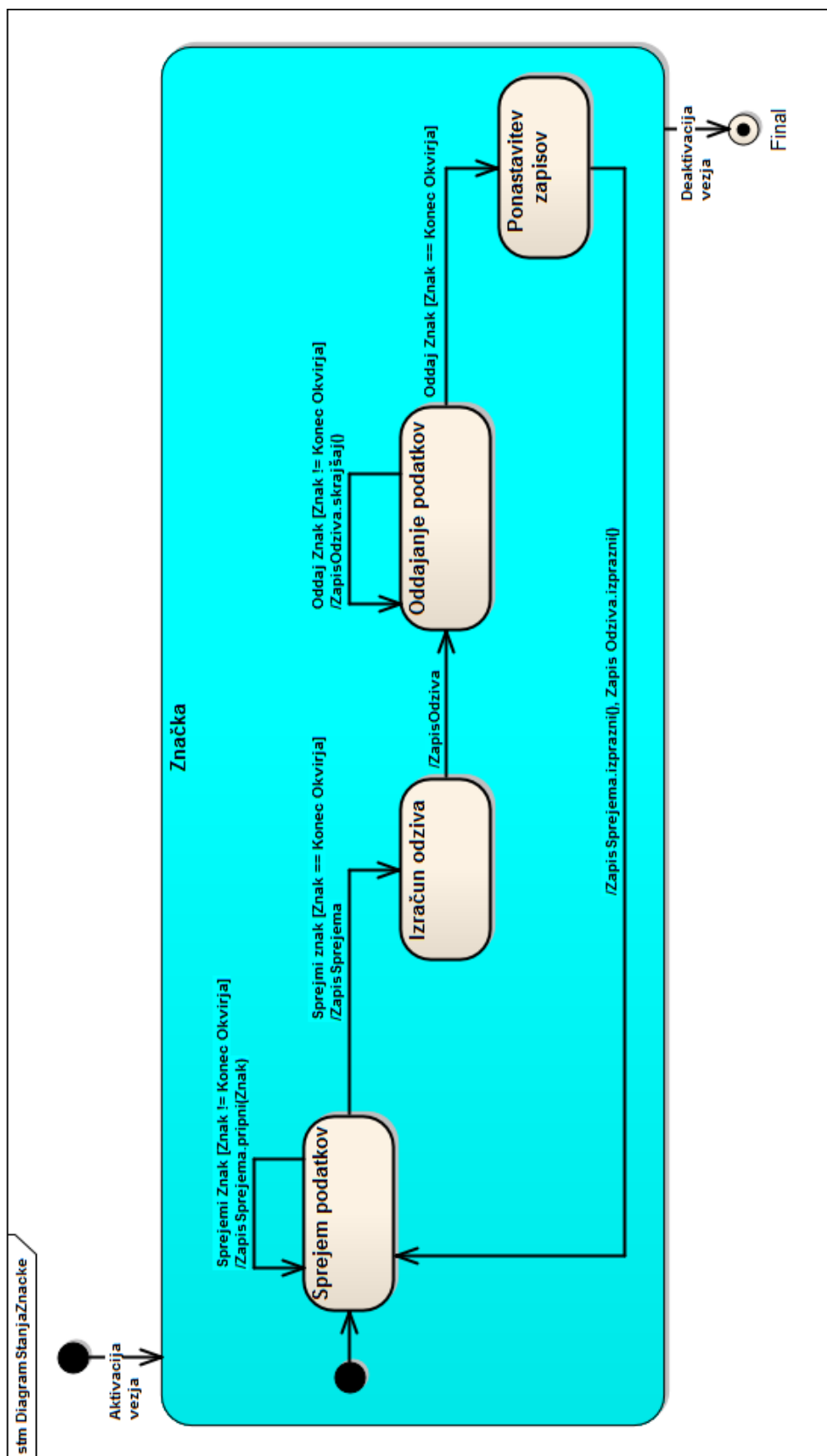
Slika 3.6: Emulacijsko okolje in binarni aplikacijski vmesnik.

Za demonstracijski sistem je realizirana značka, katere diagram stanj se nahaja na sliki 3.7. Značka se aktivira ob dogodku napajanja mikrovalovne energije. Sledi prehod v stanje sprejema podatkov. Ko značka prebere okvir, se zgodi prehod v stanje izračuna odziva. Zapisani odziv v registru se uporabi v stanju oddajanja podatkov. Po tem ko je oddan zadnji znak odziva, sledi ponastavitev registrov značke in prehod v stanje sprejema podatkov. Izvajanje značke se konča ob dogodku deaktivacije vezja.

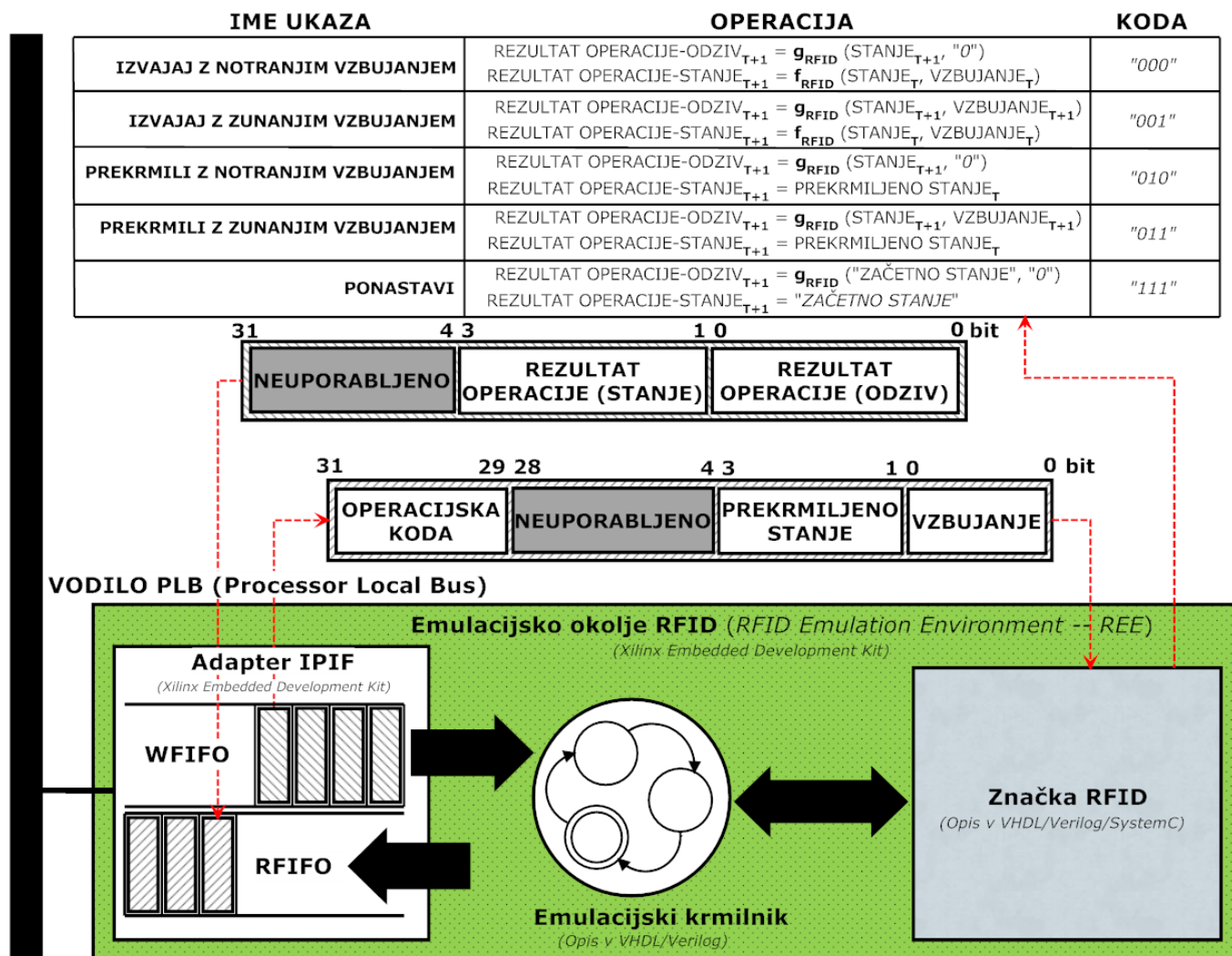
Za demonstracijski sistem in zgoraj opisano značko je vzpostavljen emulacijski sistem. Binarni aplikacijski vmesnik emulacijskega okolja je prikazan na sliki 3.8. Dolžina ukaza je 32 bitov, od tega je dolžina operacijske kode 3 bite in definira 5 ukazov.

- “Izvajaj z notranjim vzbujanjem”: Naslednje stanje je odvisno od trenutnega stanja značke in notranjega vzbujanja (vrednost 0).
- “Izvajaj z zunanjim vzbujanjem”: Naslednje stanje je odvisno od trenutnega stanja značke in zunanjega vzbujanja (drugi operand).
- “Prekrmili z notranjim vzbujanjem”: Naslednje stanje je prekrmiljeno (prvi operand). Uporabi se notranje vzbujanje značke (vrednost 0).
- “Prekrmili z zunanjim vzbujanjem”: Naslednje stanje je prekrmiljeno (prvi operand). Uporabi se zunanje vzbujanje značke (drugi operand).
- “Ponastavi”: Ponastavitev značke.

Dolžina prvega operanda, v katerem je podana vrednost prekrmiljenega stanja, je 4 bite. Dolžina drugega operanda, v katerem je podana vrednost serijskega vhoda radijskega sprejemnika značke, je 1 bit.



Slika 3.7: Diagram stanja značke.



Slika 3.8: Binarni aplikacijski vmesnik demonstracijskega sistema.

3.2.3 Komunikacijski tunel značke

Protokol komunikacijski tunel značke neposredno podpira funkcionalnost orodja za preizkušanje varnosti in nudi nadzorno upravljalске storitve emulacijskega okolja.

Osnovna naloga tunela je oddaljeno povezovanje z emulacijskim okoljem, prenos ukazov in pridobitev rezultatov operacij. Protokol tunela je realiziran s protokolom RPC [47] in definiran v jeziku RPCL [80]. Definicija protokola se nahaja v izpisu 3.1. Format sporočil protokola je skladen z binarnim aplikacijskim vmesnikom emulacijskega okolja (glej sliko 3.8).

Komunikacijski tunel zagotavlja združljivo predstavitev podatkov med heterogenimi sistemi s standardom za predstavitev, kodiranjem, prenosom in dekodiranjem podatkov XDR [48][†]. Zanesljiv prenos podatkov je zagotovljen z uporabo protokola TCP.

Uporaba protokola RPC za realizacijo protokola Komunikacijski tunel značke zahteva podporno storitev RPCBIND [49]. RPCBIND skupaj s storitvijo DNS zagotavlja uporabniško transparentnost lokacije, saj preizkuševalec dostopa do storitve na podlagi imena storitve in imena vozlišča. Povedano z drugimi besedami, preizkuševalcu ni potrebno vedeti naslova IP in številke vrat storitve.

```

1  /*!
2  * \brief REETunnel.x Opis protokola RPC.REETunnel
3  */
4
5  /* Podatkovni tipi */
6  struct RetREESat {
7      int RFIDStat; /* Status emulacijskega okolja značke RFID */
8      int DDStat;   /* Status strežnika komunikacijskega kanala */
9  };
10
11 struct RetREEOpStat {
12     unsigned int RFIDOp; /* Rezultat operacije emulacijskega okolja */
13     unsigned int DDOp;   /* Rezultat operacije strežnika komunikacijskega kanala */
14     int RFIDStat;        /* Status emulacijskega okolja značke RFID */
15     int DDStat;          /* Status strežnika komunikacijskega kanala */
16 };
17
18 struct ArgREEOp1 {
19     unsigned int DDNum; /* Ukaz strežnika komunikacijskega kanala */
20 };
21
22 struct ArgREEOp2 {
23     unsigned int DDNum; /* Ukaz strežnika komunikacijskega kanala */
24     unsigned int RFIDInstr; /* Ukaz emulacijskega okolja */
25 };
26
27 /* Procedure */
28 program REETunnel {
29     version BASIC {
30         RetREESat reset(ArgREEOp1) = 1; /* P1: Ponastavi emulacijsko okolje →
31             ↪ */
32         RetREESat write(ArgREEOp2) = 2; /* P2: Posreduje sporočilo emulacijskemu →
33             ↪ okolju */
34         RetREEOpStat read(ArgREEOp1) = 3; /* P3: Beri oddano sporočilo emulacijskega →
35             ↪ okolja */
36         RetREEOpStat getStatus(ArgREEOp1) = 4; /* P4: Ugotovi stanje emulacijskega okolja →
37             ↪ */

```

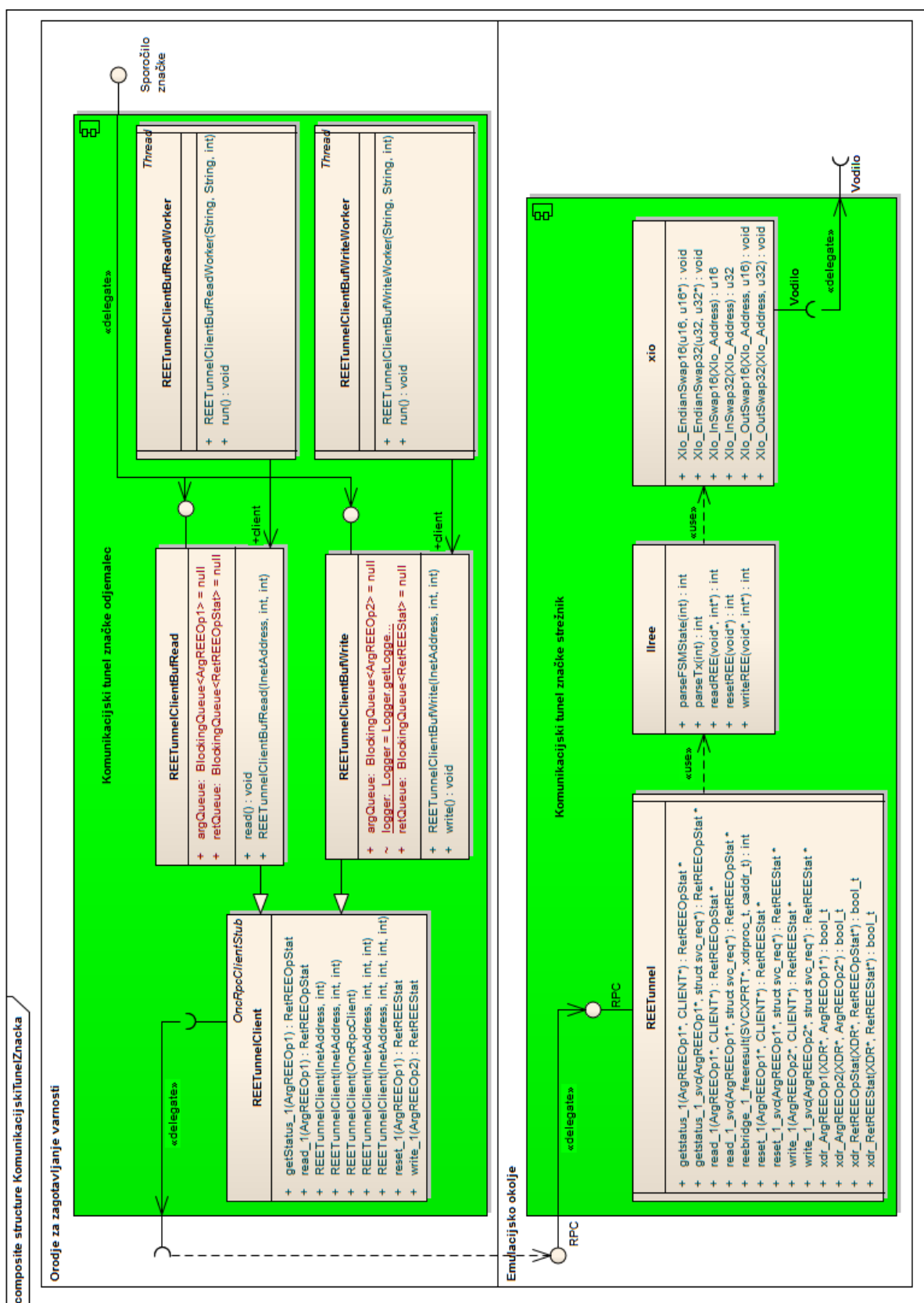
[†]Standard ASN.1. je analogni XDR. Razlika med obema je v deklaraciji tipov. Slednji ima tipe implicitno deklarirane, kar ga naredi lažjega za uporabo, a je po drugi strani manj razširljiv.

```
34     } = 1; /* Različica protokola RPC.REETunnel */  
35 } = 0x31234568; /* Enolična oznaka protokola RPC.REETunnel */
```

Izpis 3.1: Opis protokola Komunikacijskega tunela značke.

Na sliki 3.9 je prikazana struktura odjemalca in strežnika, ki realizirata zgoraj opisani protokol. Komunikacija med odjemalcem in strežnikom je sinhrona, zato je klic storitve blokirajoč. Na ta način se ohrani vrstni red izdanih ukazov v primeru izrednega dogodka izpada strežnika emulacijskega okolja ali nezanesljive mrežne storitve. Negativna posledica blokirajočega klica je čakanje aplikacije odjemalca, da se klic oddaljene procedure konča. To čakanje omejimo z uporabo izravnalnika, ki jo implementira sinhronizirana vrsta FIFO.

Naloga strežnika je prenos ukaza in branje rezultata operacij emulacijskega okolja. Strežnik prenese ukaz, tako da shrani ukaz na pomnilniško preslikano napravo emulacijskega okolja s pomočjo posebnega gonilnika. Branje rezultata operacij poteka z branjem naslova naprave emulacijskega okolja. Rezultat predstavlja stanje značke RFID in njenih izhodov v določenem časovnem trenutku. V nadaljevanju je prebrani rezultat operacije enkapsuliran in vrnjen odjemalcu.



Slika 3.9: Diagram strukture strežnika in odjemalca komunikacijskega mostu čitalca.

3.2.4 Podporna oprema komunikacijskega tunela značke

Strežnik komunikacijskega tunela značke in emulacijsko okolje sta tesno povezana preko vodila računalnika. Pri njuni komunikaciji je potrebno v prvi vrsti usklajevati različne vire okolja in reševati konfliktne situacije. V drugi vrsti pa je lepo, če okolje ponuja prijazno uporabniško okolje za razvoj in postavitve opreme.

Načrtovalska odločitev je uporaba vgrajenega sistema Xilinx MicroBlaze [81]. Ta računalnik se, podobno kot značka, sintetizira in uporabi na vezju FPGA. Uporabljeni računalnik MicroBlaze ima naslednje lastnosti:

- Arhitektura RISC.
- 32-bitni računalnik.
- 3-operandni registrsko-registrski (“load/store”) računalnik.
- 32-bitni pomnilniški naslov.
- 32-bitno podatkovno vodilo.
- Dolžina pomnilniške besede je 8 bitov.
- Vsebuje 32 splošnonamenskih 32-bitnih registrov.
- Dolžina pomnilniških operandov je 8, 16 in 32 bitov.
- Sestavljeni pomnilniški operandi so shranjeni po pravilu debelega konca.
- Vsi ukazi so dolgi 32 bitov.
- Izkorišča paralelizem na ravni ukazov in ima 5-stopenjsko cevovodno CPE.
- V eni urini periodi prevzame in izstavi največ 1 ukaz.
- Ima ukazni in podatkovni predpomnilnik. Nima navideznega pomnilnika.

Razvoj računalnika MicroBlaze poteka v razvojnem okolju Xilinx EDK. V istem razvojnem okolju je dodano tudi emulacijsko okolje kot posebna strojna naprava. Bločni diagram računalnika in sistema MicroBlaze je prikazan na sliki 3.10. Izkoriščena je tudi strojna oprema razvojnega okolja Xilinx XUP Virtex-II Pro. Razvojno okolje EDK ima pripravljene strojne module za krmilnik dostopa do prenosnega medija MAC, krmilnik pomnilnika DDR SDRAM in napravo UART.

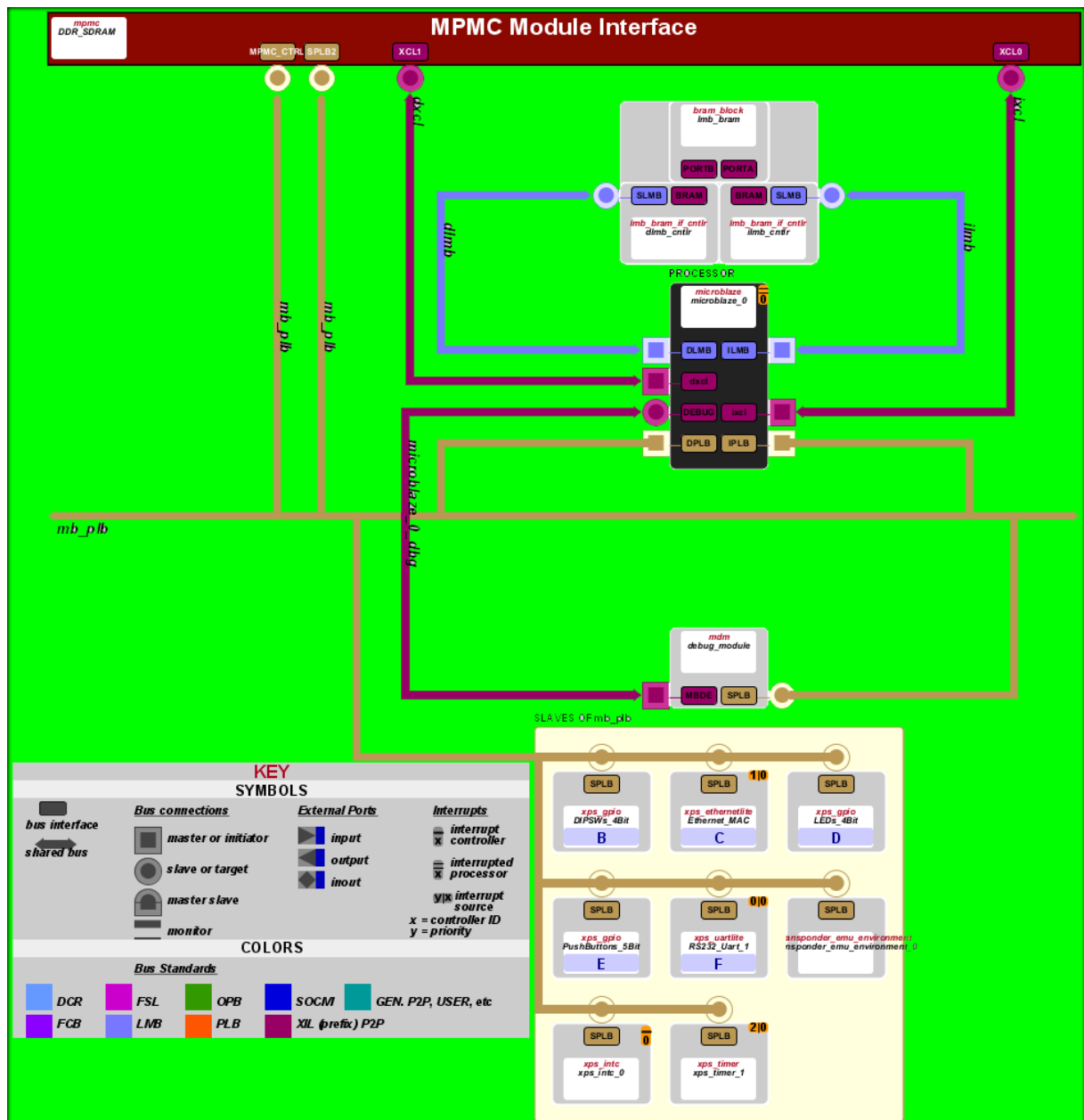
Vgrajeni sistem MicroBlaze je dovolj zmogljiv za izvajanje operacijskega sistema PetaLinux [82]. Operacijski sistem PetaLinux je osnovan na znanem operacijskem sistemu za vgrajene sisteme uClinux [83] in vsebuje dodatne programe, ki poenostavijo načrtovanje in razvoj sistema. Zlasti avtomatizirana preslikava strojnih parametrov MicroBlaze v sistem PetaLinux prihrani znaten del časa načrtovanja sistema. Sistem MicroBlaze ne

vsebuje podpore navideznega pomnilnika, zato je podpora dinamičnega povezovanja uporabniških programov težko izvedljiva. Razvojni sistem PetaLinux ponuja knjižnico za ta namen prirejene uporabniške programske opreme.

Vgrajeni sistem MicroBlaze in operacijski sistem PetaLinux omogočata

- uporabo pripravljene uporabniške programske opreme (npr. RPCBIND in Telnet),
- načrtovanje in razvoj strežnika komunikacijski tunel značke v višjem programskem jeziku C, Perl ali Python,
- podporo standardnih vmesnikov vtičnic za internetne protokole,
- multiprogramiranje in preklapljanje poslov ter
- podporo systemske programske opreme pri upravljanju medprocesne komunikacije (gonilniki protokolov, gonilniki omrežnih naprav in izravnalniki).
- upravljanje medprocesne komunikacije (npr. upravljanje protokolov TCP/IP, UART in shranjevanje sporočil v izravnalniku v jedru operacijskega sistema).

Predstavljena strojna oprema računalnika MicroBlaze in operacijski sistem PetaLinux zagotavljata podporo pri razvoju in povezovanju emulacijskega okolja RFID s strežnikom Komunikacijskega mostu značke.



Slika 3.10: Bločni diagram sistema MicroBlaze z dodanim emulacijskim okoljem.

3.2.5 Komunikacijski tunel čitalca

Protokol komunikacijski tunel čitalca neposredno podpira funkcionalnost orodja za preizkušanje varnosti in nudi nadzorno upravljalne storitve komunikacijskega strežnika čitalca.

Osnovna naloga tunela je oddaljeno povezovanje s komunikacijskim strežnikom čitalca, prenos ukazov in pridobitev rezultatov operacij. Protokol tunela je realiziran s protokolom RPC in definiran v jeziku RPCL. Definicija protokola se nahaja v izpisu 3.2. Format sporočil protokola je skladen s formatom sporočila čitalca.

Lastnosti protokola so podobne protokolu Komunikacijski tunel značke (glej poglavje 3.2.3).

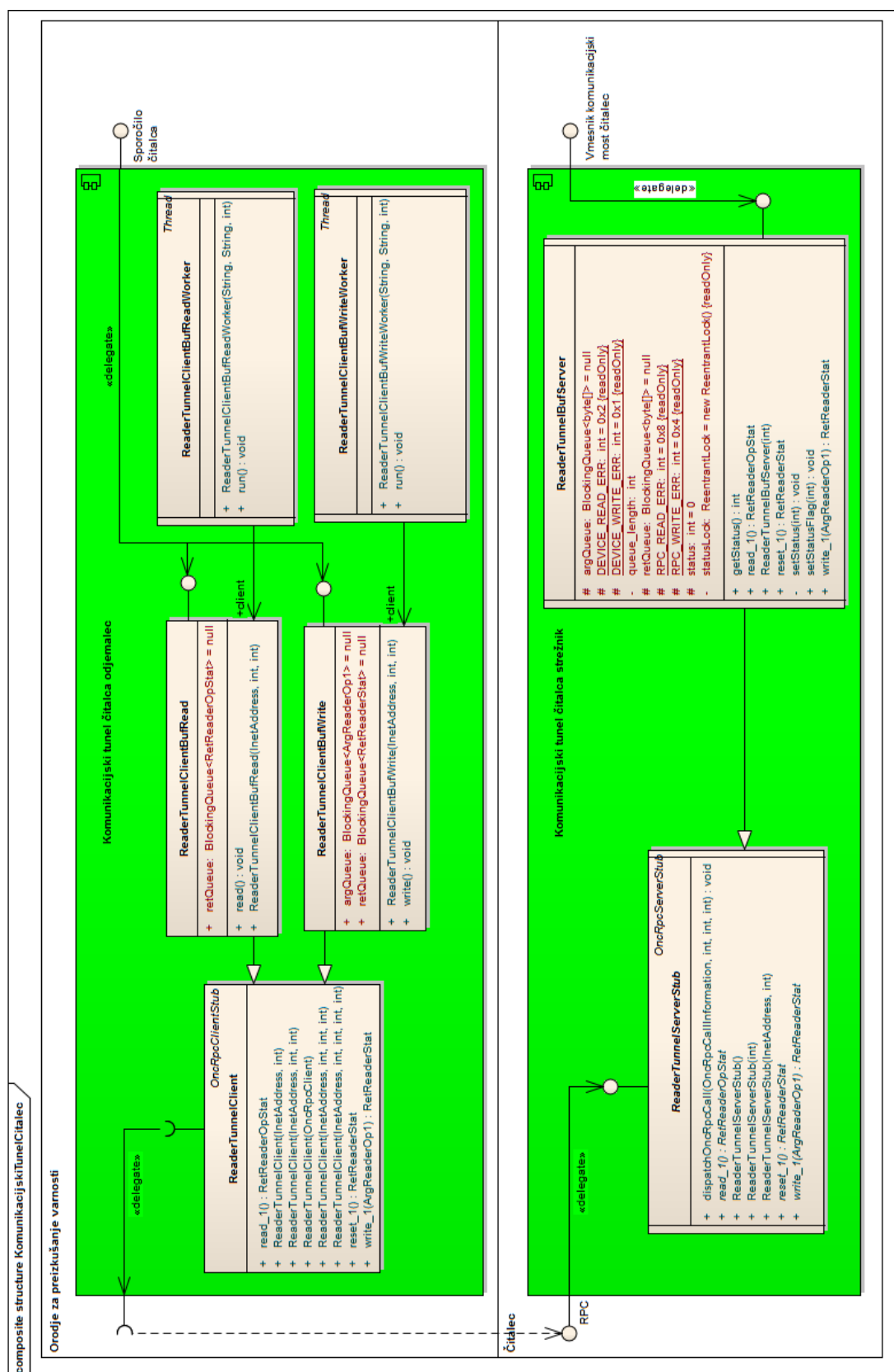
Naloga strežnika je shranjevanje in izstavljanje sporočil, ki čakajo na prenos do značke. Sporočila se do prevzema nahajajo v sinhronizirani sporočilni vrsti. Podobno velja za sporočila od značke, ki čakajo na prevzem čitalca.

```

1  /*!
2  * \brief ReaderTunel.x Opis protokola RPC.ReaderBridge
3  */
4
5  /* Podatkovni tipi */
6  struct RetReaderStat {
7      int stat; /* Status čitalca (strežnik komunikacijskega mostu) */
8  };
9
10 struct RetReaderOpStat {
11     int stat; /* Status čitalca (strežnik komunikacijskega mostu) */
12     char msg<256>; /* Oddano sporočilo čitalca, dinamična dolžina sporočila */
13     int msgByteLength; /* Dolžina sporočila (omejena na 256 zlogov) */
14 };
15
16 struct ArgReaderOp1 {
17     char msg<256>; /* Sporočilo za čitalec */
18     int msgByteLength; /* Dolžina sporočila (omejena na 256 zlogov) */
19 };
20
21 /* Procedure */
22 program READERTUNEL {
23     version BASIC {
24         RetReaderStat reset(void) = 1; /* P1: Ponastavi čitalec */
25         RetReaderOpStat read(void) = 2; /* P2: Beri oddano sporočilo čitalca */
26         RetReaderStat write(ArgReaderOp1) = 3; /* P3: Posreduj sporočilo čitalcu */
27     } = 1; /* Različica protokola RPC.ReaderTunel */
28 } = 0x31234569; /* Enolična oznaka protokola RPC.ReaderTunel */

```

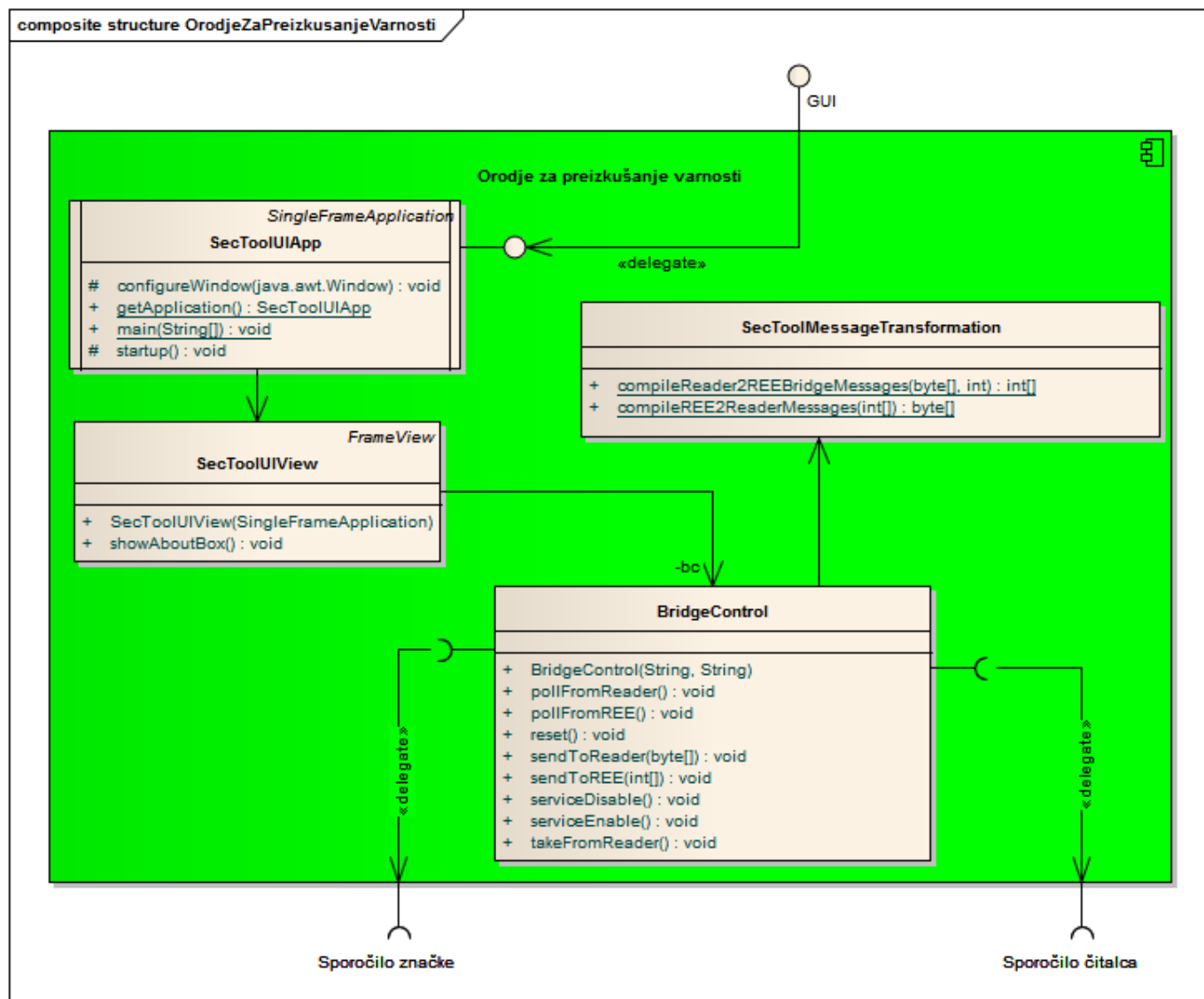
Izpis 3.2: Opis protokola Komunikacijski tunel čitalca v jeziku RPCL.



Slika 3.11: Diagram strukture strežnika in odjemalca komunikacijskega tunela čitalca.

3.2.6 Orodje za preizkušanje varnosti

Orodje za preizkušanje varnosti (glej sliko 3.12) omogoča preizkuševalcu dostop do storitev omrežnega sistema RFID. Grafični uporabniški vmesnik pri tem uporablja nadzorne funkcije protokolnega konverterja (na diag. "BridgeControl"), ta pa v nadaljevanju zahteva storitve protokola komunikacijski tunel čitalca in protokola komunikacijski tunel značke. Naloga protokolnega konverterja je transformacija sporočila čitalca v sporočilo značke in obratno.

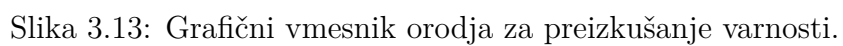


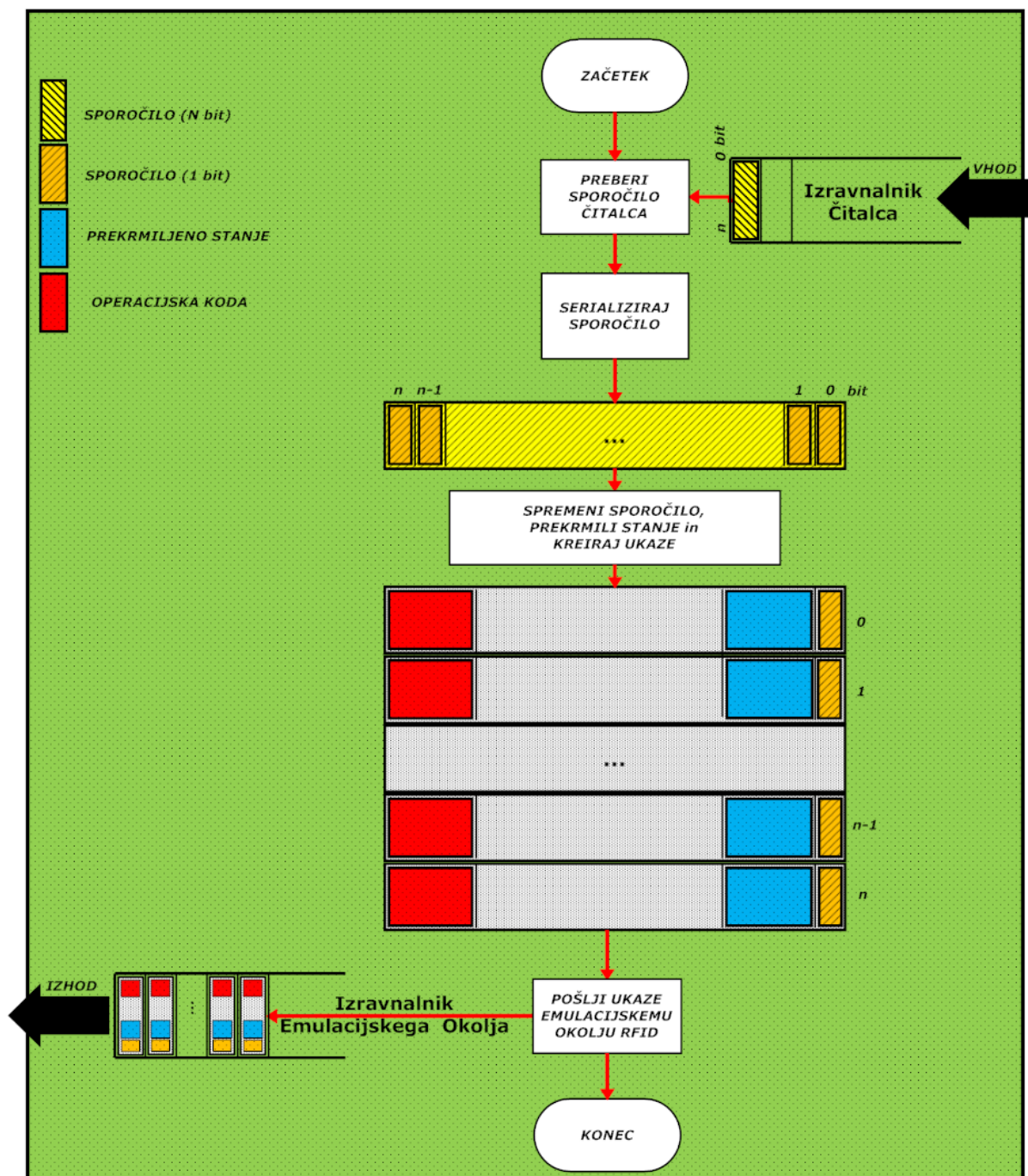
Slika 3.12: Diagram strukture orodja za preizkušanje varnosti.

Uporaba grafičnega vmesnika orodja za preizkušanje varnosti je prikazana na sliki 3.13. Zaporedna številka reference na sliki je enaka zaporedni številki spodnjega pojasnila.

1. Preizkuševalec preveri, ali je čitalec poslal novo sporočilo.

2. Novo sporočilo N_1 je dodano v prvo tabelo in je modro označeno. Preizkuševalec lahko spremeni vrednost sporočila čitalca.
3. Preizkuševalec transformira sporočilo čitalca. Za vsak bit sporočila je ustvarjen nov ukaz, ki vsebuje operacijsko kodo, vrednost prekrmljenega stanja in n -ti bit sporočila. Potek transformacije sporočila čitalca je prikazan na sliki 3.14. Transformacija je skladna z aplikacijskim binarnim vmesnikom na sliki 3.8.
4. Transformirano sporočilo je dodano v drugo tabelo. Sestavljajo ga novi ukazi, ki so označeni z modro barvo in dobijo zaporedno številko $\#N_1, N_2$. Vrednost $\#N_1$ je referenca na prvo tabelo. Privzeti ukazi ne prekrmlijo stanja značke. Preizkuševalec lahko spremeni operacijsko kodo, vrednost prekrmljenega stanja in n -ti bit sporočila.
5. Preizkuševalec pošlje modro označene ukaze emulacijskemu okolju.
6. Preizkuševalec preveri, ali so rezultati operacij emulacijskega okolja na voljo.
7. Rezultati operacij emulacijskega okolja so dodani v tretjo tabelo in dobijo zaporedno številko N_3 . Preizkuševalec lahko spremeni vrnjeno vrednost odziva (in tudi vrnjeno vrednost stanja).
8. Preizkuševalec transformira modro označene vrednosti odziva v sporočilo čitalca.
9. Novo sporočilo je dodano v četrto tabelo in dobi zaporedni številki $\#N_{3.S}, \#N_{3.E}$, ki označujeta začetno in končno zaporedno številko vrnutih vrednosti, ki so se upoštevale pri transformaciji sporočila.
10. Preizkuševalec pošlje modro označeno sporočilo nazaj čitalcu.





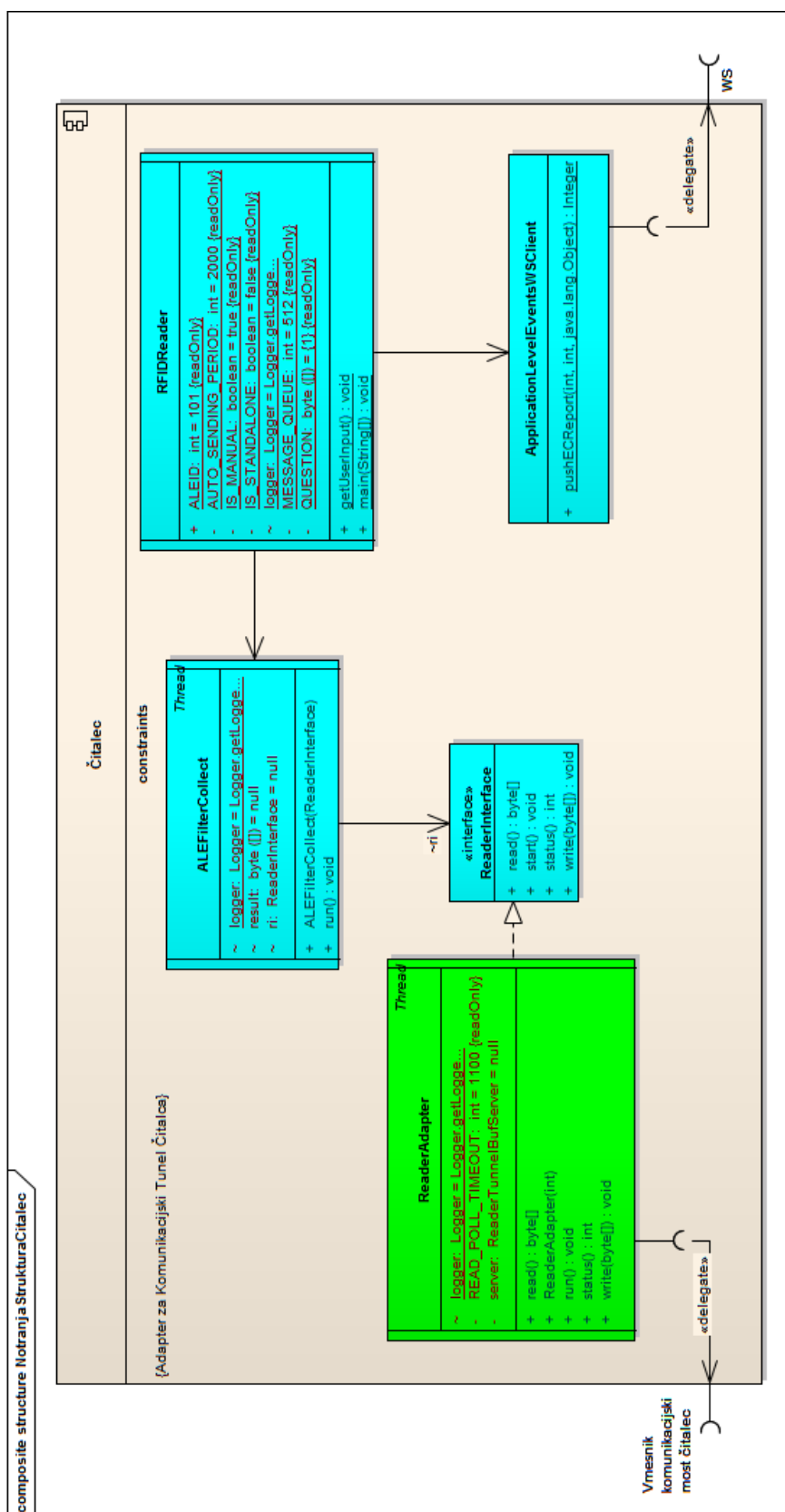
Slika 3.14: Diagram poteka transformacije sporočila čitalca.

3.2.7 Čitalec

Naloga čitalca je komunikacija z značkami, ki se nahajajo v dometu signala antene. Čitalec izzove značko in počaka na njen odgovor – elektronsko kodo izdelka (EPC). Če ima značka še druge podatke poleg kode EPC, je omogočeno tudi njihovo branje. V nekaterih primerih je mogoče celo podatke spreminjati. Čitalec lahko dodatno zahteva še izvedbo drugih storitev, odvisno od funkcionalnosti značke. Na primer uničenje značke, zaklep značke itn. Te storitve se nahajajo na aplikacijski plasti značke.

Druga naloga čitalca je filtriranje in agregacija prejetih podatkov. Na ta način čitalec ustvari zapise dogodkov na aplikacijski ravni abstrakcije (it. angl. Application Level Events). Ti podatki so primerni za nadaljnjo obravnavo na procesni, proizvodni ali poslovni plasti informacijskega sistema. Naloga čitalca je posredovati prepoznani dogodek naprej v zaledni del sistema. Čitalec sporoča dogodke z odjemalcem spletne storitve preko protokola SOAP. Opis spletne storitve je opisan v jeziku WSDL in se nahaja v izpisu 3.3.

Struktura čitalca je predstavljena na sliki 3.15. Čitalec v osnovni različici bere in pošilja podatke na brezžično napravo. V okviru orodja je potrebno vmesnik pisanja in branja s komunikacijske naprave zamenjati s posebnim adapterjem. Ta adapter povezuje vmesnik čitalca z vmesnikom komunikacijskega strežnika čitalca, ki realizira protokol komunikacijski tunel čitalca za povezovanje z orodjem za preizkušanje varnosti.



Slika 3.15: Bločni diagram notranje strukture čitalca.

```

1  <?xml version='1.0' encoding='UTF-8'?>
2  <!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RIs version is Metro/2.1 ( →
   ↪ branches/2.1-6728; 2011-02-03T14:14:58+0000) JAXWS-RI/2.2.3 JAXWS/2.2. -->
3  <definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity- →
   ↪ utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wspl_2="http:// →
   ↪ schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/ →
   ↪ addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns=" →
   ↪ http://ws.epcis/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas →
   ↪ .xmlsoap.org/wsdl/" targetNamespace="http://ws.epcis/" name=" →
   ↪ ApplicationLevelEventsWS">
4  <types>
5  <xsd:schema>
6      <xs:element name="pushECReport" type="tns:pushECReport" />
7      <xs:element name="pushECReportResponse" type="tns:pushECReportResponse" />
8      <xs:complexType name="pushECReport">
9          <xs:sequence>
10             <xs:element name="epc" type="xs:int" />
11             <xs:element name="aleid" type="xs:int" />
12             <xs:element name="eventDate" type="xs:anySimpleType" minOccurs="0" />
13          </xs:sequence>
14      </xs:complexType>
15      <xs:complexType name="pushECReportResponse">
16          <xs:sequence>
17             <xs:element name="return" type="xs:int" minOccurs="0" />
18          </xs:sequence>
19      </xs:complexType>
20  </xsd:schema>
21  </types>
22  <message name="pushECReport">
23      <part name="parameters" element="tns:pushECReport" />
24  </message>
25  <message name="pushECReportResponse">
26      <part name="parameters" element="tns:pushECReportResponse" />
27  </message>
28  <portType name="ApplicationLevelEventsWS">
29      <operation name="pushECReport">
30          <input wsam:Action="http://ws.epcis/ApplicationLevelEventsWS/ →
   ↪ pushECReportRequest" message="tns:pushECReport" />
31          <output wsam:Action="http://ws.epcis/ApplicationLevelEventsWS/ →
   ↪ pushECReportResponse" message="tns:pushECReportResponse" />
32      </operation>
33  </portType>
34  <binding name="ApplicationLevelEventsWSPortBinding" type="tns:ApplicationLevelEventsWS">
35      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
36      <operation name="pushECReport">
37          <soap:operation soapAction="" />
38          <input>
39              <soap:body use="literal" />
40          </input>
41          <output>
42              <soap:body use="literal" />
43          </output>
44      </operation>
45  </binding>
46  <service name="ApplicationLevelEventsWS">
47      <port name="ApplicationLevelEventsWSPort" binding=" →
   ↪ tns:ApplicationLevelEventsWSPortBinding">
48          <soap:address location="http://sol.fri.uni-lj.si:8080/EPCIS/ →
   ↪ ApplicationLevelEventsWS" />
49      </port>
50  </service>
51  </definitions>

```

Izpis 3.3: Opis spletne storitve za prenos dogodka elektronske kode v jeziku WSDL.

3.2.8 Zaledni del sistema

Naloga zalednega dela sistema je prepoznavanje poslovnih dogodkov, povezanih z značko RFID, in njihova nadaljnja obravnava za hranjenje, iskanje, obdelavo in prikazovanje podatkov. Zaledni del sistema sestavljajo sistem za upravljanje s podatkovno bazo, sistem za zajetje dogodkov in sistem za prikaz dogodkov.

Končni uporabnik dostopa do storitev sistema za prikaz dogodkov, ki nudi storitev informacijske podpore za odločanje in podporo delovnih procesov združbe. Prepoznavanje poslovnih dogodkov in shranjevanje je storitev sistema za zajetje dogodkov. Sistem za upravljanje s podatkovno bazo nudi učinkovito storitev kreiranja, vzdrževanja in nadzora nad dostopom (iskanje in obdelava) do podatkov.

Diagram postavitve orodja za zagotavljanje varnosti, ki vključuje tudi zaledni del sistema, je predstavljen na sliki 3.16. Na diagramu so predstavljeni vsi artifakti, ki so potrebni za postavitev okolja. V nadaljevanju bo sledil opis vseh treh komponent zalednega dela sistema.

Sistem za prikaz dogodkov

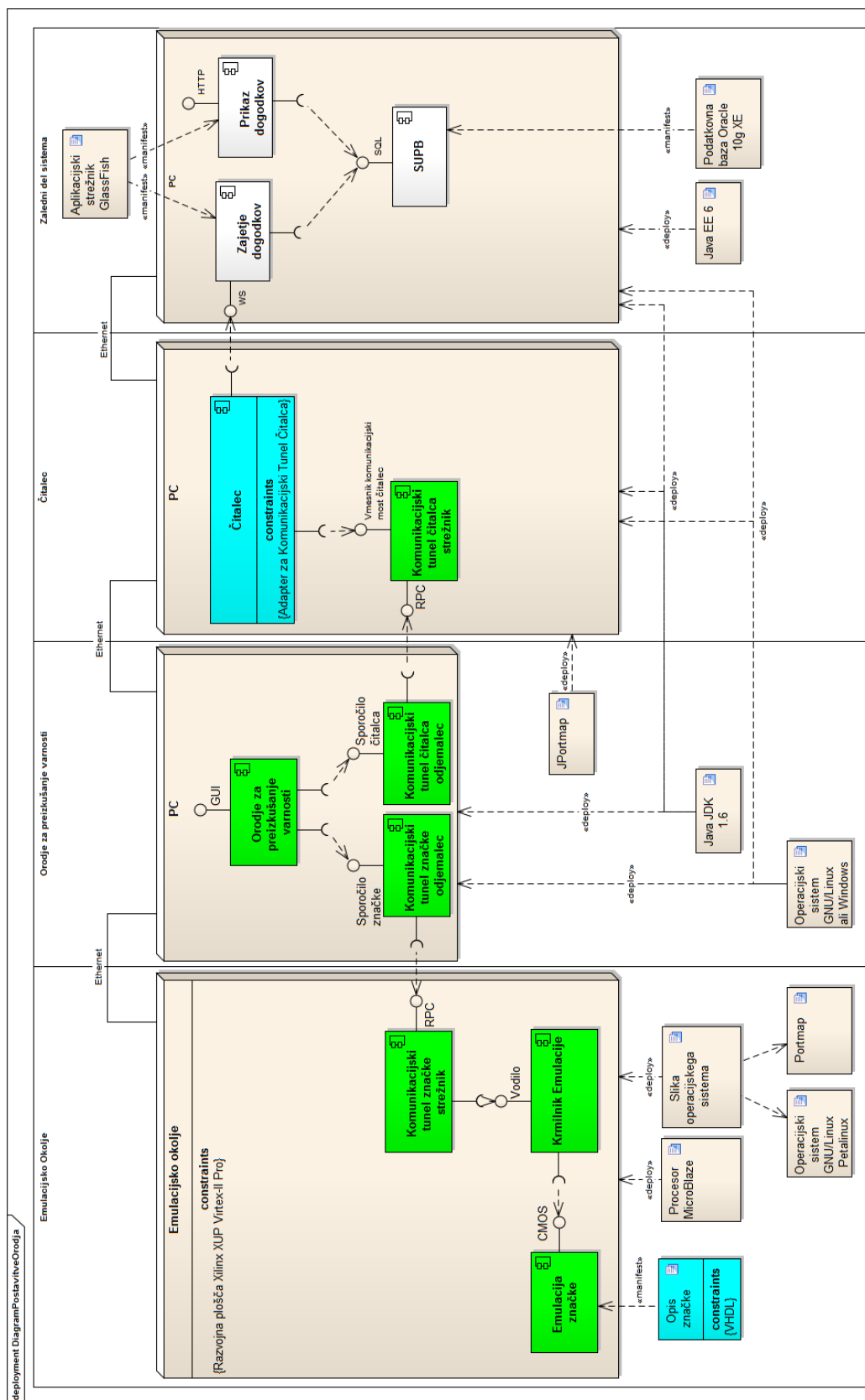
Demonstracijski sistem za prikaz dogodkov je spletna aplikacija na strežniku GlassFish. Le-ta je vsebnik za aplikacije, ki temeljijo na tehnologiji Java EE. Storitve sistema za prikaz dogodkov je prikaz dnevnika dostopa do sistema za nadzor fizičnega dostopa in prikaz statistike dostopov in zavrnitev. Aplikacija je spletna stran HTML, ki je dostopna preko protokola HTTP (glej sliko 3.17).

Sistem za upravljanje s podatkovno bazo

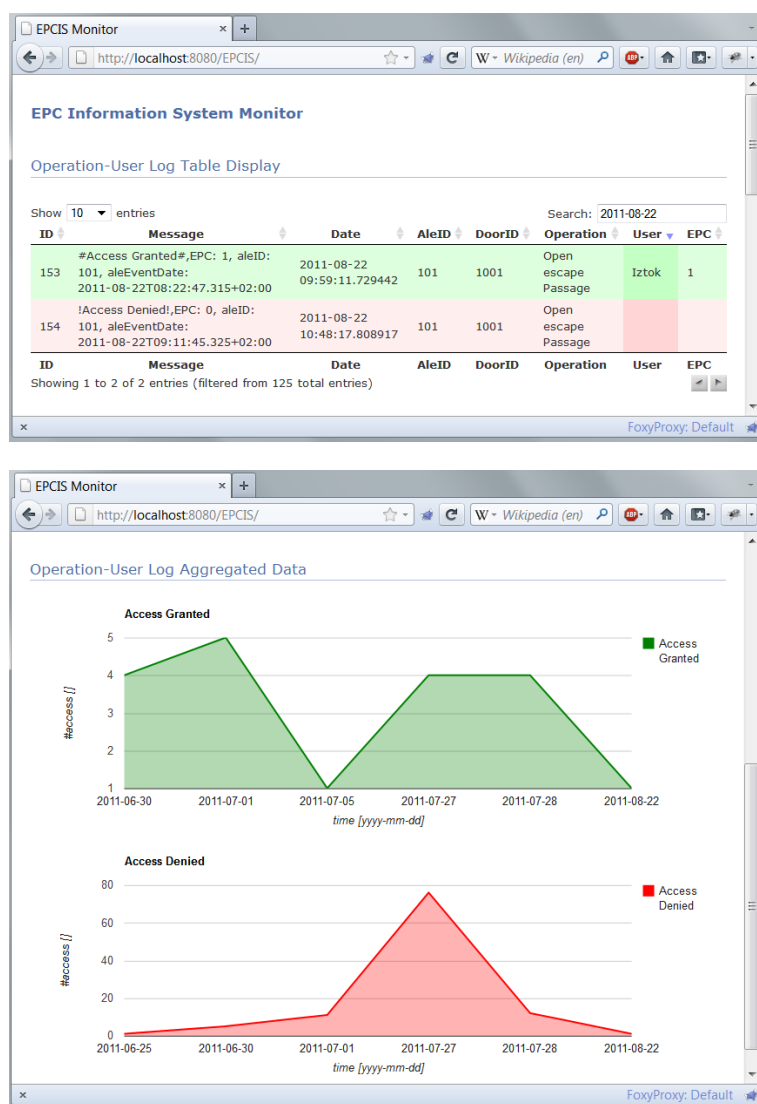
Konceptualni in pripadajoči logični podatkovni model sistema nadzora dostopa je prikazan na sliki 3.18 in 3.19. Diagrama predstavljata model nadzora dostopa na podlagi vlog. Vsak zaposleni dobi brezžično značko, ki vsebuje elektronsko kodo produkta. Zaposleni pristopi k prehodu in približa značko čitalcu. Čitalec posreduje elektronsko kodo produkta in identifikacijsko številko čitalca zalednemu delu sistema. Ta overi in avtorizira uporabnika, tj. preveri identiteto nosilca značke in preveri, ali ima zaposleni dodeljeno vlogo, ki omogoča izvedbo operacije, npr. odklep vrat.

Vsakemu zaposlenemu subjektu je dodeljena vloga. Vsaka vloga je avtorizirana za določene operacije. Vloga ima hierarhično sestavo. To pomeni, da subjekt s specializirano vlogo lahko dostopa do operacij generalizirane vloge.

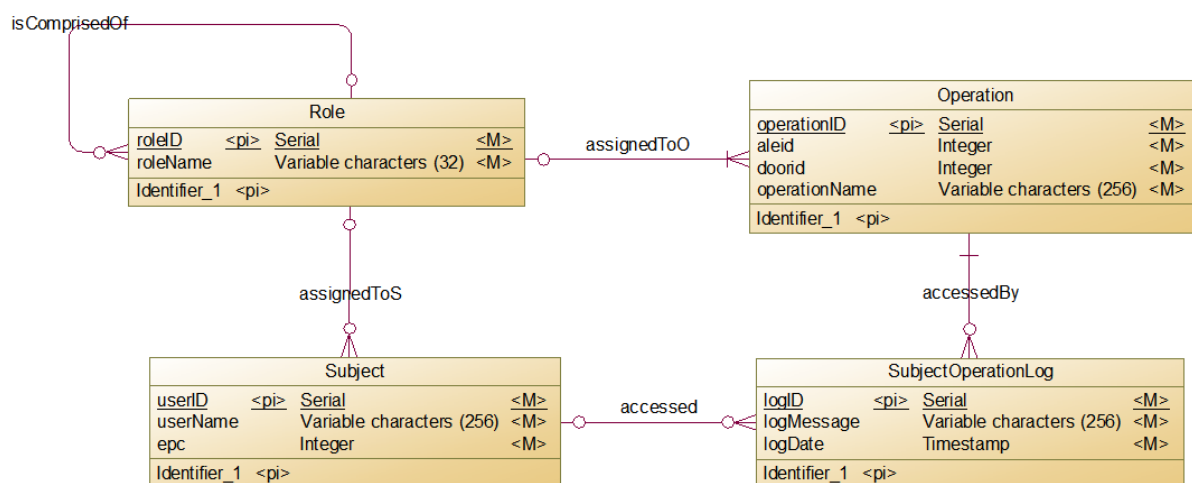
Avtorizacija operacije poteka, tako da sistem poišče seznam dodeljenih in hierarhičnih vlog uporabnika na podlagi podane elektronske kode izdelka. Nato preveri, ali te vloge omogočajo izvedbo zahtevane operacije, ki jo označuje identifikacijska številka čitalca. Ta transakcija je prikazana na izpisu 3.4 kot prva poizvedba.



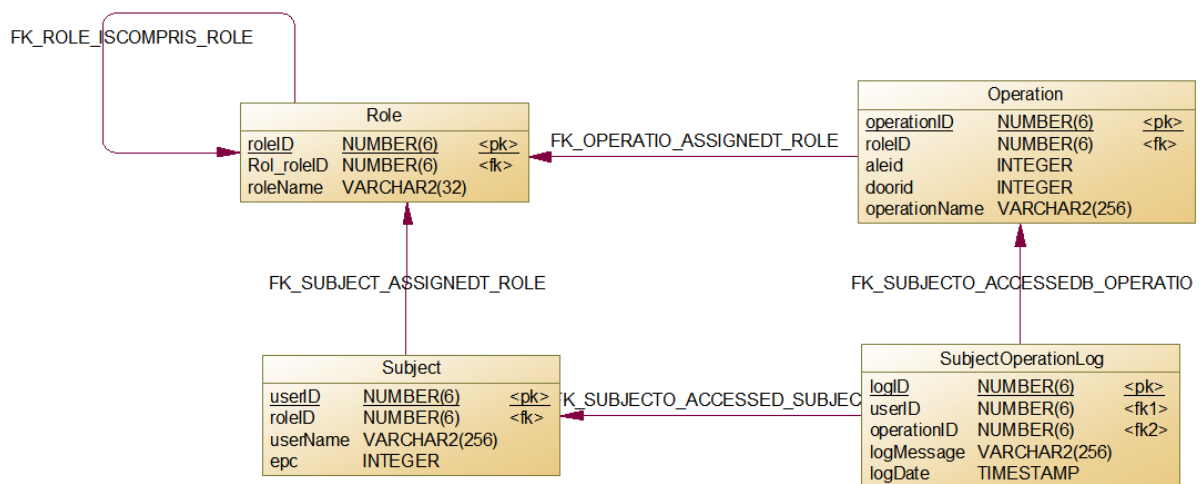
Slika 3.16: Diagram postavitve orodja.



Slika 3.17: Prikaz delovanja spletne strani sistema za prikaz dogodkov.



Slika 3.18: Konceptualni podatkovni model.



Slika 3.19: Logični podatkovni model.

```

1  /*
2  * POIZVEDBA 1:
3  * Overi uporabnika na podlagi podane kode EPC in
4  * identifikatorja čitalca ALEID. Poizvedba upošteva
5  * hiarhijo vlog.
6  */
7  SELECT s.userID, op.operationID, s.userName, op.doorid
8  FROM Operation op, Role r, Subject s
9  WHERE r.roleID = s.roleID
10     AND op.roleID IN /* Vektor vlog uporabnika določen na podlagi */
11                      /* identifikatorja kode EPC (Hiearhična poizvedba) */
12                      (SELECT r.roleID
13                       FROM Role r
14                       /* Koren hiearhije */
15                       START WITH r.roleID = /* Doloci trenutno vozlisce */
16                       /* na podlagi kode EPC */
17                       (SELECT r.roleID
18                        FROM Role r, Subject s
19                        WHERE s.roleID = r.roleID
20                        /* I. Parameter EPC */
21                        AND s.epc = ?)
22                       /* Povezava med vozliščem očeta in sina */
23                       CONNECT BY PRIOR r.Rol_roleID = r.roleID)
24  /* II. Parameter ALEID */
25  AND op.aleid = ?
26  /* III. Parameter EPC */
27  AND s.epc = ?;
28
29  /*
30  * POIZVEDBA 2:
31  * Določi identifikator uporabnika userid na podlagi
32  * kode EPC.
33  */
34  SELECT s.userID, s.userName
35  FROM Subject s
36  WHERE epc = ?; /* I. Parameter EPC */
37
38  /*
39  * POIZVEDBA 3:
40  * Določi identifikator operacije na podlagi
41  * identifikatorja ALEID.
42  */
43  SELECT op.operationID, op.doorid
44  FROM Operation op
45  WHERE aleid = ?; /* I. Parameter ALEID */
46
47  /*
48  * POIZVEDBA 4:
49  * Vnos v dnevnik
50  */
51  INSERT INTO SubjectOperationLog(logID, userID, operationID, logDate, logMessage)
52  VALUES (S_SUBJECTOPERATIONLOG.nextval, /* Proženje sekvence za določitev */
53          /* identifikatorja primarnega ključa dnevnika */
54          ?, /* I. Paramter userID */
55          ?, /* II. Paramter operationID */
56          CURRENT_TIMESTAMP, /* Časovni žig */
57          ?); /* III. Parameter sporočilo dnevnika */
58
59  /*
60  * POIZVEDBA 5:
61  * Izpis celotnega dnevnika urejenega po datumu,
62  * pripadajoče ime operacije za vsak vnos in
63  * ime uporabnika za vsak vnos, če le-ta obstaja.
64  */
65  SELECT sol.logID AS logID, sol.logMessage AS logMessage, sol.logDate AS logDate,
66         op.aleid AS aleid, op.doorid AS doorid, op.operationName AS operationName,
67         s.userName AS userName, s.epc AS epc
68  FROM SubjectOperationLog sol JOIN Operation op ON sol.operationID = op.operationID
69         LEFT OUTER JOIN Subject s ON sol.userID = s.userID

```

```

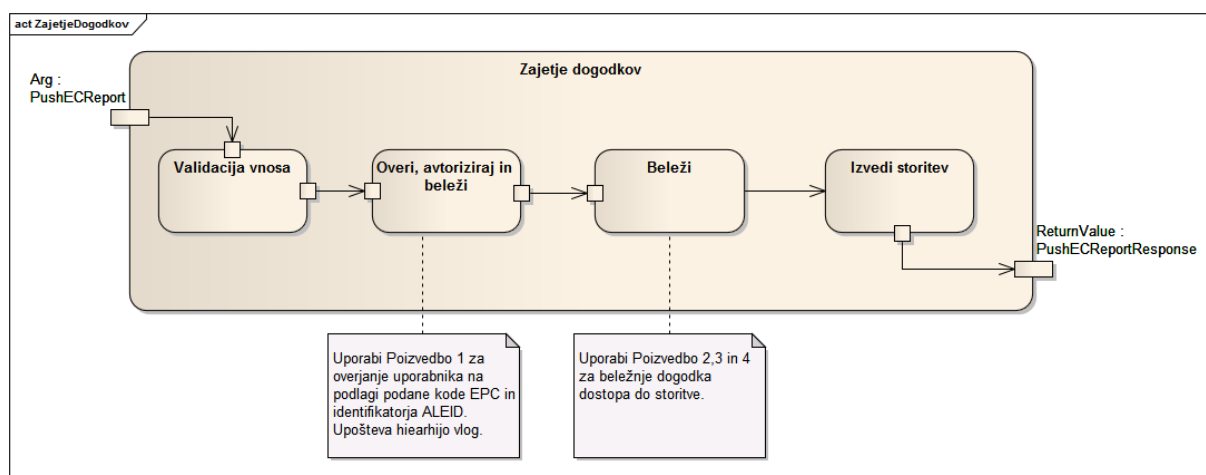
70 ORDER BY sol.logDate DESC;
71
72 /*
73  * POIZVEDBA 6:
74  * Izpis celotnega dnevnika urejenega po datumu
75  * za zavrnjene dostope
76  */
77 SELECT TO_CHAR(sol.logDate, 'yyyy-mm-dd') AS logDate, COUNT(*) AS adnum
78 FROM SubjectOperationLog sol
79 WHERE sol.logMessage LIKE '%Access_Denied%'
80 GROUP BY TO_CHAR(sol.logDate, 'yyyy-mm-dd')
81 ORDER BY logDate ASC;
82
83 /*
84  * POIZVEDBA 7:
85  * Izpis celotnega dnevnika urejenega po datumu
86  * za omogočene dostope
87  */
88 SELECT TO_CHAR(sol.logDate, 'yyyy-mm-dd') AS logDate, COUNT(*) AS agnum
89 FROM SubjectOperationLog sol
90 WHERE sol.logMessage LIKE '%Access_Granted%'
91 GROUP BY TO_CHAR(sol.logDate, 'yyyy-mm-dd')
92 ORDER BY logDate ASC;

```

Izpis 3.4: Poizvedbe SQL uporabljene pri zajetju in prikazu dogodkov.

Sistem za zajetje dogodkov

Avtorizacija in beleženje dogodkov je ključna storitev komponente zajetja dogodkov. Storitve se izvaja na strežniku GlassFish in zahteva uporabo protokola SOAP za prenos sporočil. Opis storitve je naveden v jeziku WSDL in se nahaja v izpisu 3.3. Diagram aktivnosti storitve se nahaja na sliki 3.20. Preko klica storitve strežnik pridobi podatek o dogodku značke in čitalca. Le-ta vsebuje elektronsko kodo produkta značke, identifikacijsko številko čitalca in časovni žig čitalca. Po sprejetju zahteve sistem validira podatke. Če so podatki ustrezni, sistem overi in avtorizira uporabnika elektronske kode produkta. Poskus dostopa do virov sistema se zabeleži. Za to se uporabijo poizvedbe izpisa 3.4 pod zaporedno številko 2, 3 in 4. Če sta overjanje in avtorizacija uspešno zaključena, se uporabniku omogoči storitev.

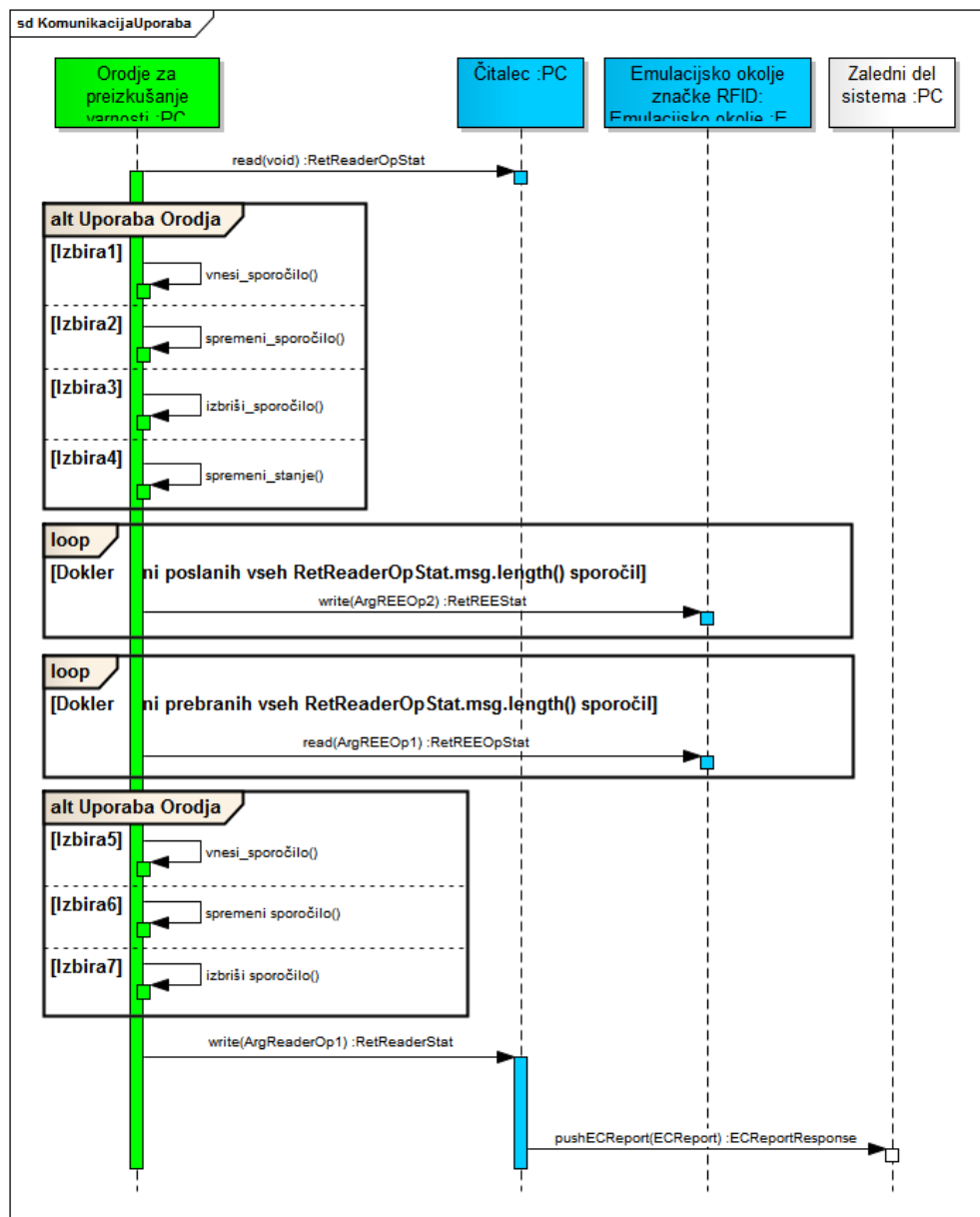


Slika 3.20: Diagram aktivnosti zajetja dogodkov.

3.3 Postopek preizkušanja varnosti

Preizkušanje varnosti poteka z orodjem v nadzorovanem okolju, ki vključuje simulacijo napada sistema na podlagi prepoznanih sredstev in groženj. Preizkuševalec realizira grožnje, napade in učinke, ki so zbrani v tabeli 2.1 in prikazani na sliki 3.1, s kombinacijo dveh napadov: (i) z napadom na komunikacijo med značko in čitalcem in (ii) z napadom na izvajanje logičnih operacij značke. Lastnosti postopka preizkušanja varnosti razvrstimo po tipologiji testiranja [50]: (i) potek testiranja je časovno sinhroniziran s ciljnim sistemom in diskreten, (ii) izbira testov je dinamična in odvisna od reakcije značke in čitalca, (iii) kriterij za izbiro testa je vnos napak v izvajanje logičnih operacij značke in njene komunikacije s čitalcem na podlagi prepoznanih groženj.

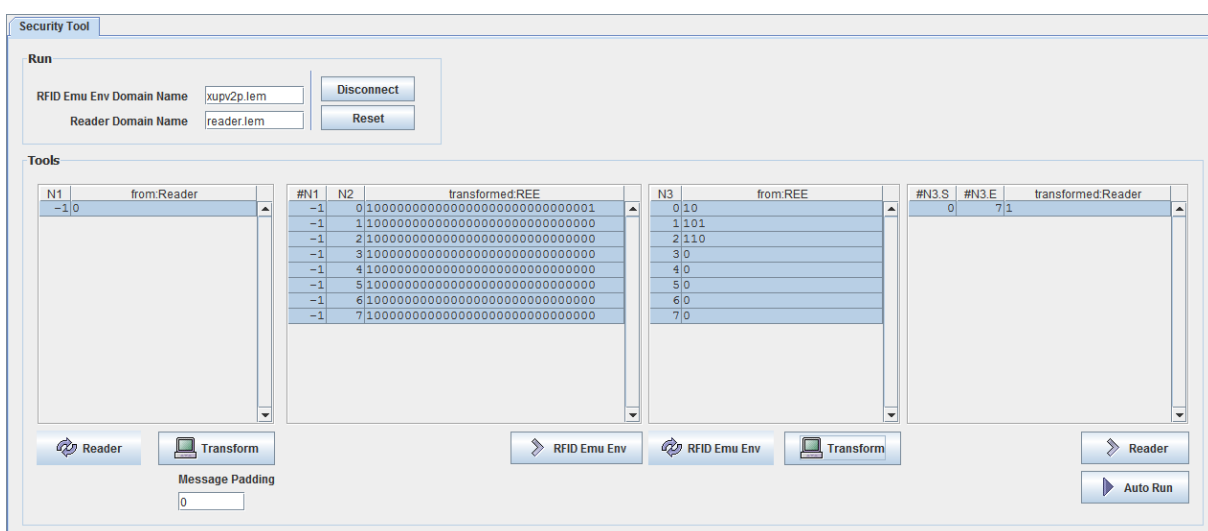
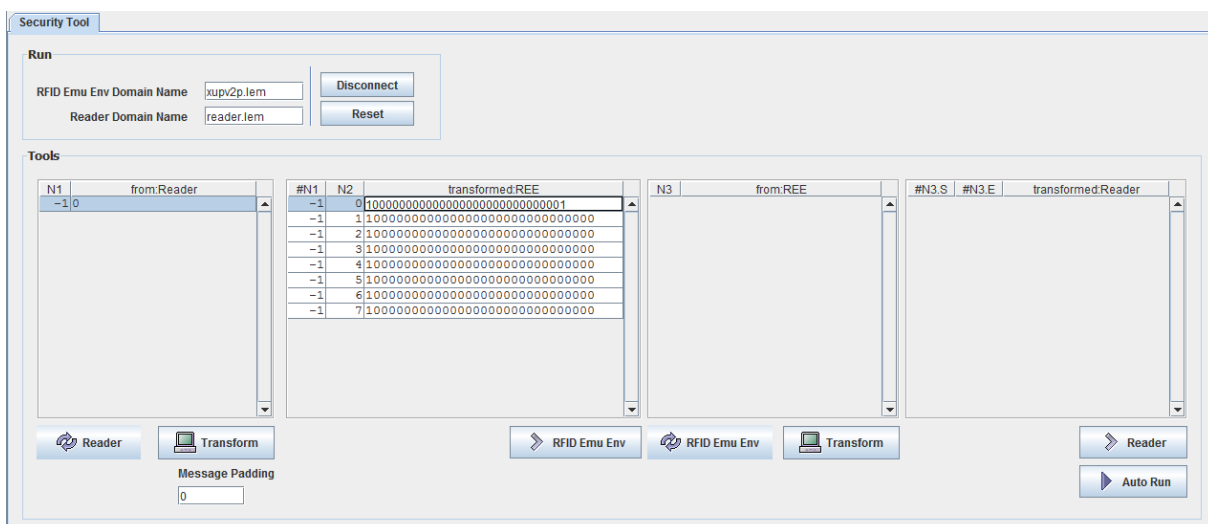
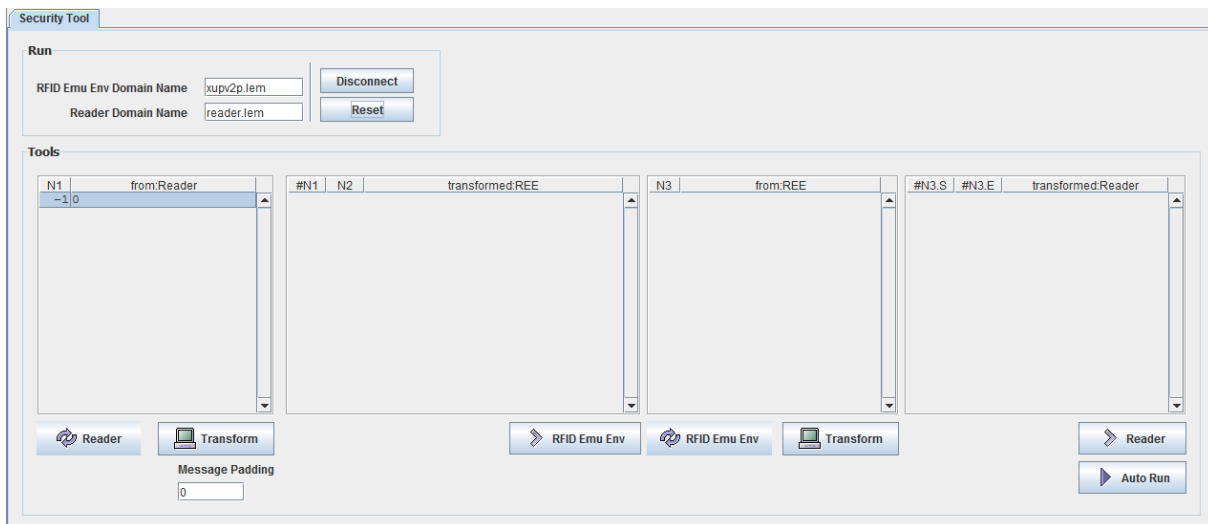
Medtem ko so bile posamezne funkcije orodja predstavljene v poglavju 3.2.6, bo to poglavje obravnavalo postopek preizkušanja. Generični postopek preizkušanja se nahaja na diagramu zaporedja 3.21.



Slika 3.21: Diagram zaporedja komunikacije sistema in uporabe orodja.

Napadalec ima na voljo sedem osnovnih možnosti pri preizkušanju, ki jih je mogoče v nadaljevanju kombinirati.

1. Napadalec vnese sporočilo čitalca (glej sliko 3.22).
2. Napadalec spremeni sporočilo čitalca (glej sliko 3.23).
3. Napadalec izbriše sporočilo čitalca (glej sliko 3.24).
4. Napadalec prekrmili stanje značke (glej sliko 3.25).
5. Napadalec vnese sporočilo značke (glej sliko 3.26).
6. Napadalec spremeni sporočilo značke (glej sliko 3.27).
7. Napadalec izbriše sporočilo značke (glej sliko 3.28).



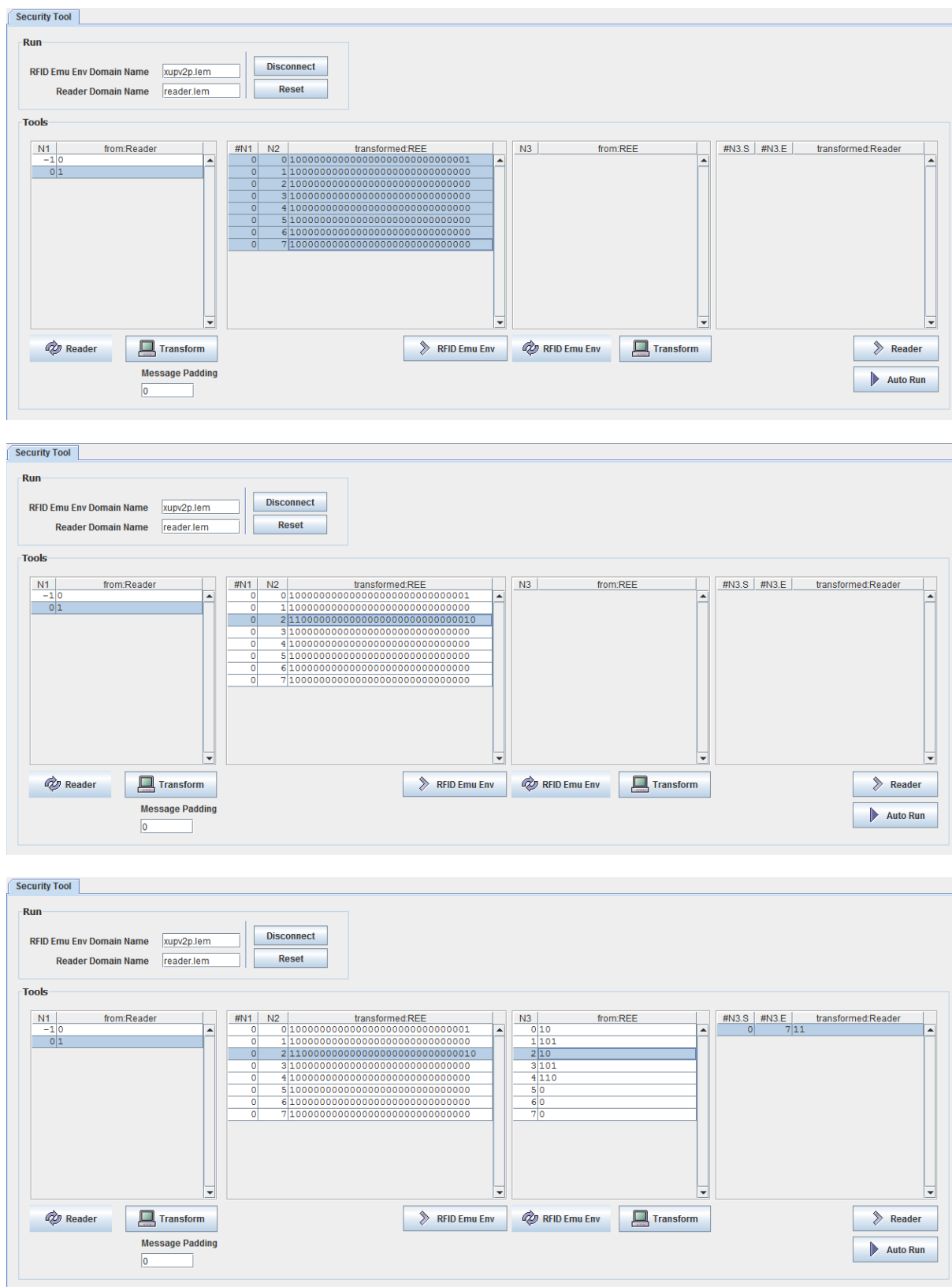
Slika 3.22: Postopek vnosa sporočila čitalca.



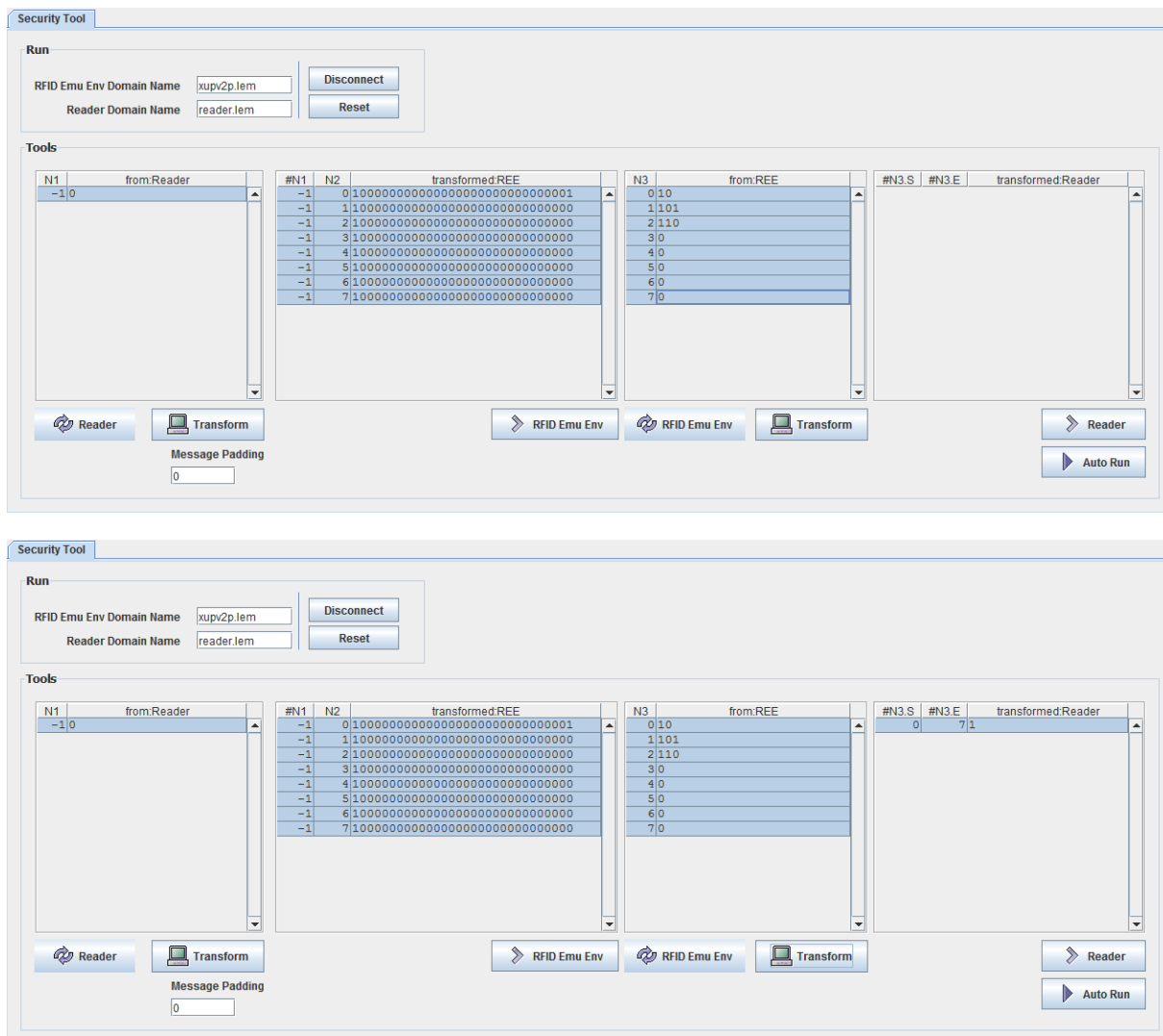
Slika 3.23: Postopek spremembe sporočila čitalca.



Slika 3.24: Postopek izbrisa sporočila čitalca.



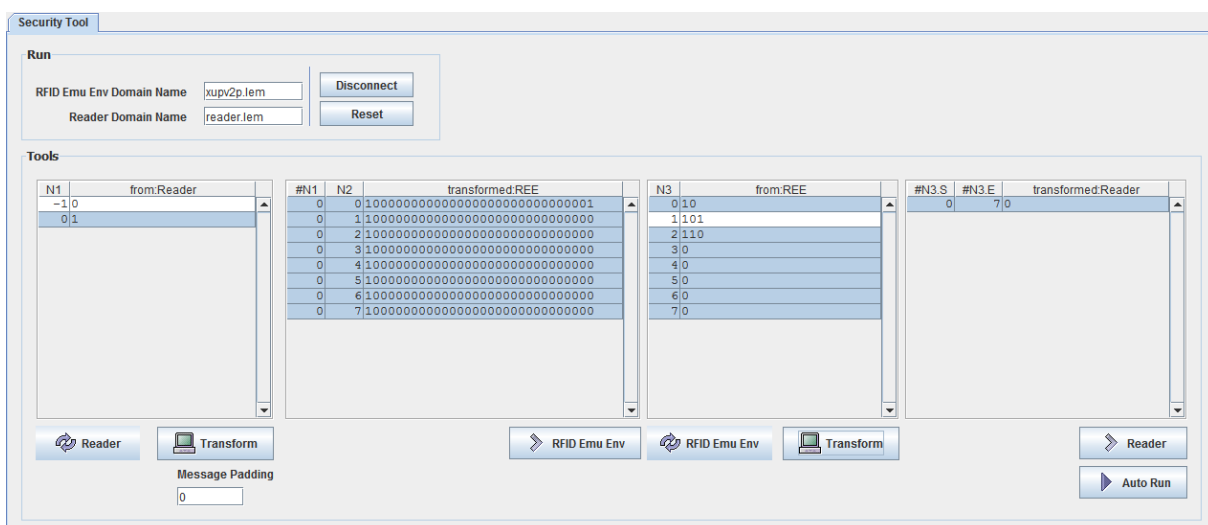
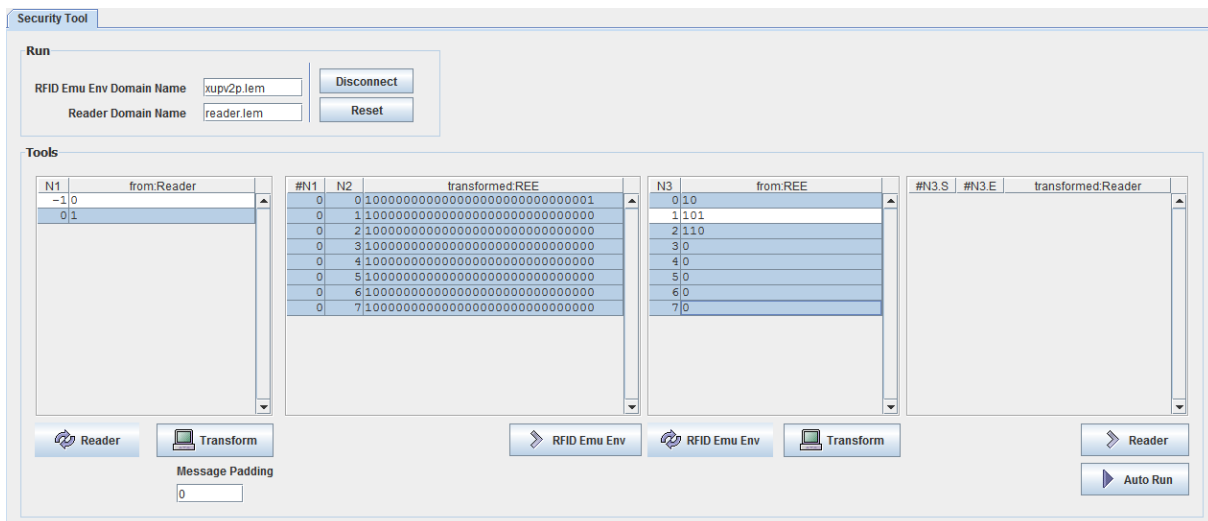
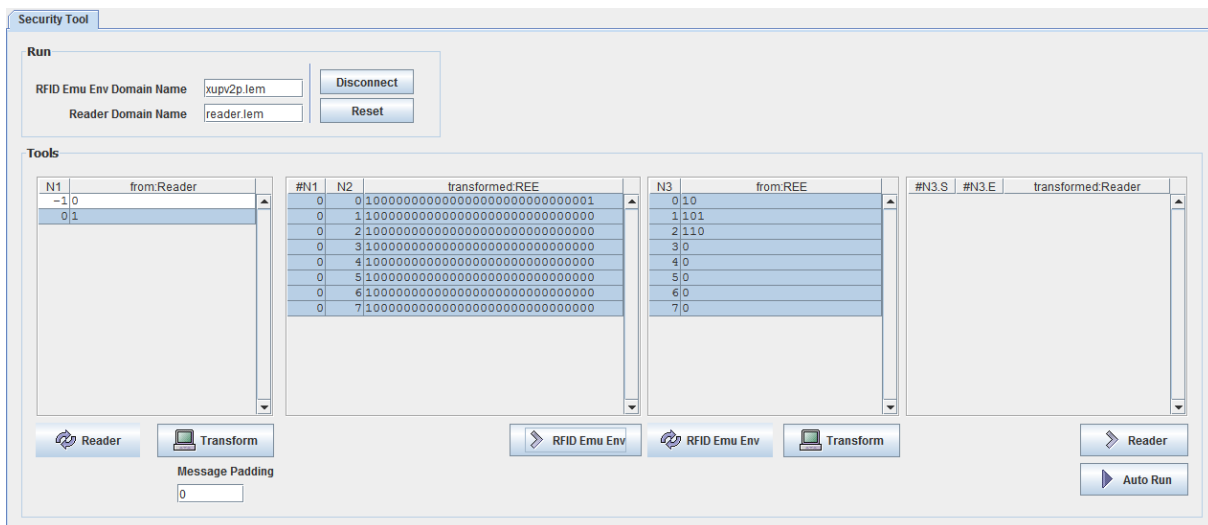
Slika 3.25: Postopek spremembe stanja značke.



Slika 3.26: Postopek vnosa sporočila značke.



Slika 3.27: Postopek spremembe sporočila značke.



Slika 3.28: Postopek izbrisa sporočila značke.

3.4 Metoda za zagotavljanje varnosti

Metoda za zagotavljanje varnosti je specializacija procesa obvladovanja tveganja ISO/IEC 27005:2008 za okolja omrežnih sistemov RFID s preizkušanjem značk RFID. Proces obvladovanja je bil predstavljen v poglavju 2.7 in se nahaja na diagramu poteka 2.5.

Najprej bo predstavljena specializacija procesa obvladovanja tveganja. V nadaljevanju bodo postopki priprave orodja, gradnje in uporabe umeščeni v ta proces. Na koncu bomo izpostavili pozitivne učinke uporabe metode.

3.4.1 Proces obvladovanja tveganja

Metoda za zagotavljanje varnosti je prikazana na diagramu poteka 3.29. Določitev konteksta predstavlja osnovo za vse nadaljnje korake, podobno kot v procesu obvladovanja tveganja ISO/IEC 27005:2008. V sklopu prepoznavanja tveganja se izvedejo naslednje aktivnosti: prepoznavanje sredstev, groženj in posledic, ki pretijo poslovnemu sistemu. Po opravljeni aktivnosti prepoznavanja obstoječih protiukrepov se prične aktivnost prepoznavanja ranljivosti, ki je razdeljena na tri faze:

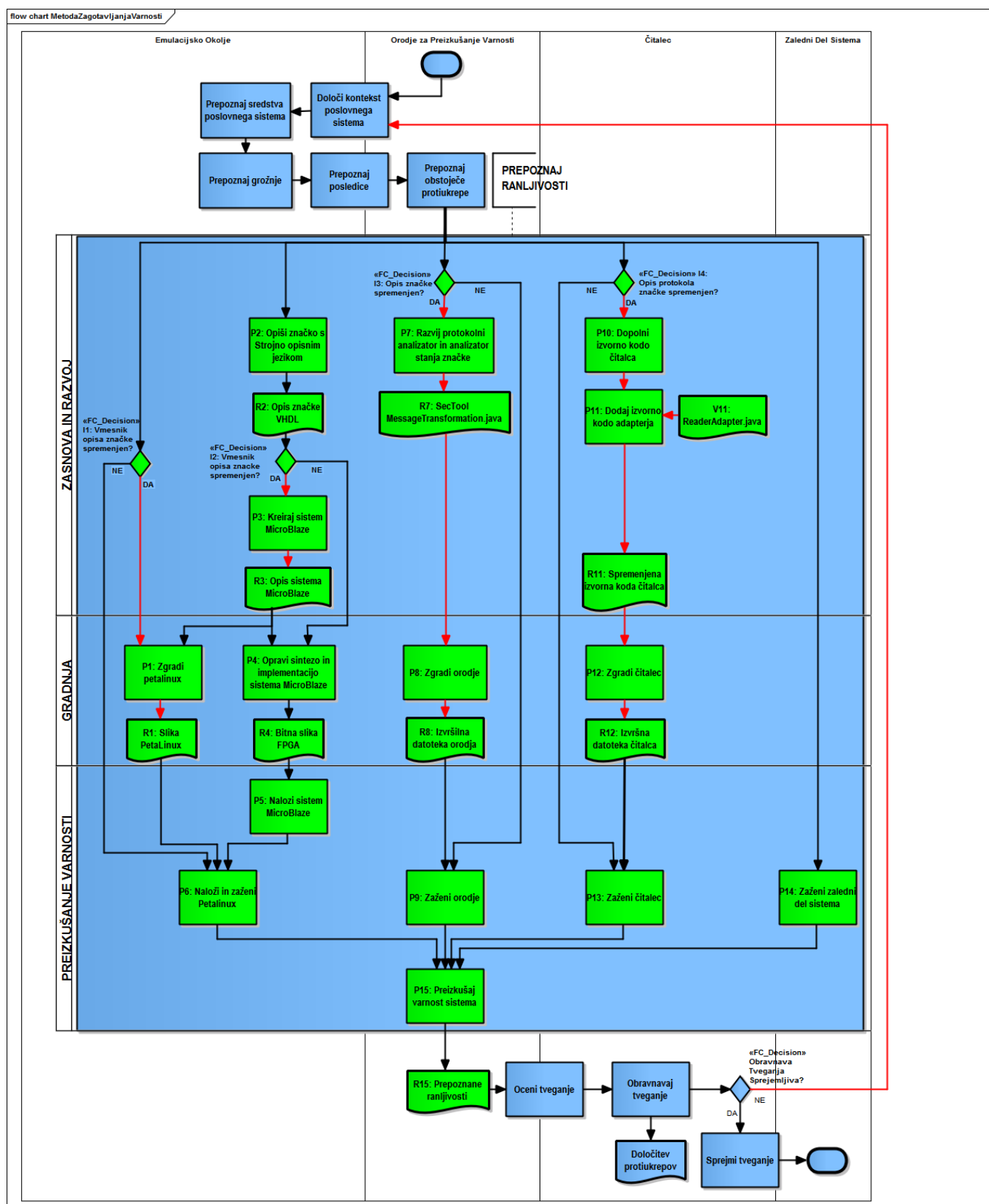
1. Zasnova in razvoj orodja za preizkušanje varnosti.
2. Gradnja orodja za preizkušanje varnosti.
3. Preizkušanje varnosti sistema.

Na podlagi prepoznanih ranljivosti je določena ocena tveganja na podlagi sprejetih meril pri določitvi konteksta. Tveganja so obravnavana, tako da se stopnja tveganja zmanjša, zadrži, tveganju se izogne ali tveganje se prenese.

- V primeru zmanjšanja tveganja je določen zadosten in učinkovit protiukrep, ki se ga doda v sistem. Za to spremembo je potrebno znova prepoznati ranljivosti. Potrebno je preveriti ali je protiukrep pravilen in učinkovit ter ali vnese nove ranljivosti v sistem. Če je ocena tveganja tokrat zadovoljiva, se tveganje sprejme in postopek se konča.
- V primeru izognitve in prenosa tveganja so izbrani primerni ukrepi obvladovanja tveganja, tako da je zadoščeno zahtevam standarda ISO/IEC 27001:2005[†].

Tveganja je potrebno redno spremljati, preverjati in obveščati pooblašcene uporabnike. V okviru prepoznavanja tveganja so lahko uporabljene tudi druge komplementarne metode, ki jih ta dokument ne navaja. Kakovost procesa obvladovanja tveganja vpliva na kakovost sistema za upravljanje varovanja informacij, ta pa na kakovost poslovnega sistema.

[†]Izbrane so lahko tudi druge metode za zagotavljanje varnosti, kot jih na primer navaja ISO/IEC TR 15443-2:2005 [51].



Slika 3.29: Diagram poteka metode za zagotavljanje varnosti za okolja omrežnih sistemov radio-frekvenčne identifikacije.

Kakovost procesa obvladovanja tveganja ima posledice za naslednje štiri akterje.

1. Deležniki so odjemalci storitev ali so kako drugače vpleteni v poslovni sistem.
2. Združbe ponujajo storitev deležnikom, upravljajo kakovost storitve in upravljajo nabavo sistema ali komponente.
3. Razvijalci načrtujejo, razvijejo in postavijo sistem ali komponento.
4. Hakerji izkoriščajo ranljivosti sistema ali komponente na nelegitimen način za osebno korist.

Storitev je izpostavljena nižjemu tveganju, če so varnostne zahteve deležnikov in združbe zagotovljene. Pri tem je pomembno, da združba varnostne zahteve prepozna in jih zahteva pri nabavi sistema ali komponente. Razvijalci morajo te zahteve upoštevati pri načrtovanju, razvoju in postavitvi sistema, če se nabava nanaša na izdelek po meri. Če se nabava nanaša na izdelek široke potrošnje, je naloga združbe, da preveri ali opisane varnostne funkcionalnosti izdelka izpolnjujejo varnostne zahteve.

Ne glede na tip nabave združbe ne smejo slepo verjeti trditvam razvijalcev, da opisane varnostne funkcionalnosti izdelka držijo v celoti. Glede na analizo in oceno tveganja storitve morajo deležniki zahtevati ustrezno stopnjo zagotavljanja varnostnih funkcionalnosti izdelka, tako da opredelijo strogost in obseg ocenjevanja varnostih funkcionalnosti. Te trditve morajo biti nato podvržene evalvaciji, ki upošteva zahteve ocenjevanja. Na podlagi rezultata evalvacije se združba odloči za nabavo ali pa odstop od nabave izdelka[†].

3.4.2 Prepoznavanje ranljivosti

Proces prepoznavanja ranljivosti (glej sliko 3.29) je razdeljen na tri faze:

1. Zasnova in razvoj orodja za preizkušanje varnosti.
2. Gradnja orodja za preizkušanje varnosti.
3. Preizkušanje varnosti sistema.

Potek prepoznavanja tveganja je odvisen od načrtovanih sprememb značke, ki izhajajo iz vnosa novih protiukrepov ali drugih sprememb logike značke. To so: sprememba vmesnika značke (binarni aplikacijski vmesnik), sprememba določenega protokola značke in ostale spremembe značke.

V prvi iteraciji so izpolnjeni vsi pogoji vozlišč (I1, I2, I3 in I4). Zasnova, razvoj in gradnja orodja je tako najbolj obsežna. Druga iteracija procesa prepoznavanja ranljivosti se

[†]Primer ogrodja za evalvacijo varnosti izdelka, kjer so specificirane zahteve varnostnih funkcionalnosti in njihove zahtevane stopnje zagotavljanja, predstavlja standard skupnih meril za evalvacijo varnosti informacijske tehnologije ISO/IEC 15408 [52].

izvede, če se v prvi prepoznajo ranljivosti. Pri obravnavanju tveganja so določeni protiu-krepi. Protiukrepe je potrebno preizkusiti, ali so pravilni in učinkoviti ter ali vnesejo novo ranljivost v sistem. Zaradi dodanih sprememb značke je potrebno posodobiti posamezne dele orodja.

Potek posodobitve orodja pri zasnovi, razvoju in gradnji določa vrsta spremembe značke.

- Pogoj I1 je izpolnjen, če se vmesnik opisa značke spremeni in je potrebno zaradi tega definirati nov binarni aplikacijski vmesnik.
- Pogoj I2 je izpolnjen, če se vmesnik opisa značke spremeni in je potrebno zaradi tega definirati nov binarni aplikacijski vmesnik.
- Pogoj I3 je izpolnjen, če sprememba opisa značke zahteva spremembo protokolnega analizatorja in analizatorja stanja značke.
- Pogoj I4 je izpolnjen, če se spremeni protokol značke.

Postopek načrtovanja posameznih aktivnosti $P_{1,...,15}$ je podrobno opisan v dodatku.

- Postopek gradnje Petalinux (P1, R1 in P6) je opisan v C. Postopek gradnje uporabniške aplikacije (velja tudi za P1, R1 in P6) je opisan v privesku D.
- Primer opisa značke, primerne za integracijo z orodjem (P2 in R2) se nahaja v privesku E.1.
- Postopek gradnje sistema MicroBlaze (P3, R3, P4, R4 in P5) je opisan v privesku B.
- Primer opisa protokolnega analizatorja (P7 in R7) se nahaja v privesku E.2.
- Izvedba vmesnika čitalca z adapterjem (V11 in P11) se nahaja v privesku E.3.
- Aktivnosti P8, R8, P9, P10, R11, P12, R12 in P14 se izvedejo po meri glede na uporabljeno tehnologijo.
- Primer preizkušanja varnosti sistema (P15) se nahaja v odseku 3.3.

3.4.3 Pozitivni učinki uporabe metode

Uporaba orodja in metode ima dva pozitivna učinka.

1. Uporaba orodja in metode pripomore k bolj učinkovitemu prepoznavanju ranljivosti v procesu obvladovanja tveganja za okolja omrežnih sistemov RFID. Aktivnost prepoznavanja ranljivosti z orodjem je stroškovno učinkovita. Prepoznavanje ranljivosti poteka že v razvoju izdelka, če gre za nabavo izdelka po meri. Pri nabavi izdelka široke potrošnje pa prepoznavanje ranljivosti poteka pri postavitvi sistema. Uporaba orodja in metode poveča možnost zgodnje prepoznavne ranljivosti, zgodnje

določitve tveganja izpostavljenega sistema in pravočasnega obravnavanja tveganja z vpeljavo primernih in učinkovitih protiukrepov. Na ta način imajo hekerji manjšo možnost, da ranljivost razkrijejo in jo začnejo izkoriščati.

2. Uporaba orodja in metode pripomore k bolj prepričljivemu zagotavljanju varnostnih funkcionalnosti. To lahko predstavlja konkurenčno prednost pred drugimi izdelki, če so zahteve združb ali deležnikov glede ocenjevanja varnostnih funkcionalnosti izdelka dovolj stroge in obsežne.

V ANALIZI nas bodo zanimale karakteristike razvoja, gradnje in uporabe orodja. Proučili bomo omejitve pri uporabi orodja. Primerjali bomo prednosti in slabosti simulacije za preizkušanje varnosti v primerjavi z emulacijo značke, ki jo podpira predstavljeno orodje. Prikaz delovanja orodja nas ne bo zanimal, saj je bil predstavljen v poglavju 3.3.

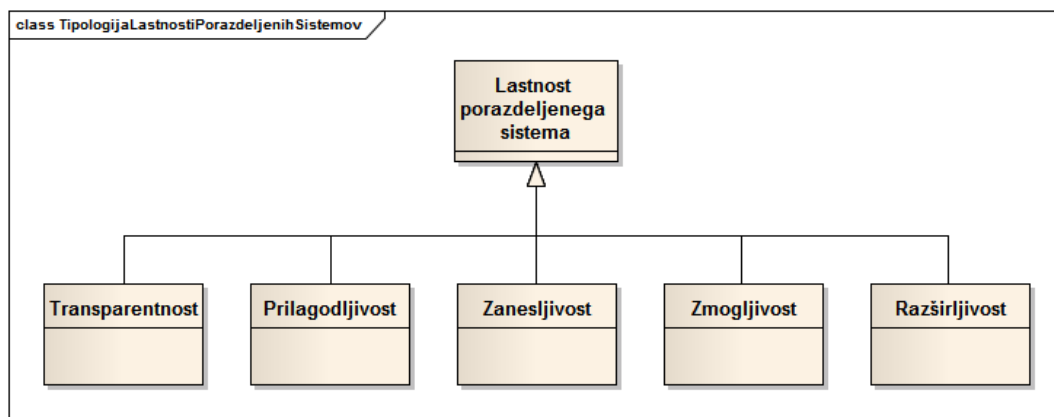
Nazadnje nas bo zanimal vpliv uporabe orodja na kakovost sistema. Pri tem se bomo opirali na pozitivne učinke uporabe orodja, ki so bili predstavljeni v poglavju 3.4.3. S simulacijo bomo proučili dinamiko obnašanja kakovosti sistema. Pri tem bomo primerjali posledice slabega in dobrega obvladovanja tveganja. Slabo obvladovanje tveganja bo predstavljalo resnično (referenčno) obnašanje, ko je obvladovanje tveganja reaktivno in kasni glede na incidente. Dobro obvladovanje tveganja bo predstavljalo izboljšanje obvladovanja tveganja z uporabo orodja.

Osnovo za gradnjo simulacijskega modela bo predstavljal diagram vzroka in posledic karakteristik kakovosti sistema (glej sliko 2.4). Za simulacijo in modeliranje bo izbrana sistemska dinamika [53], za katero obstaja računalniško podprt pristop k analizi in oblikovanju novih strategij oz. politik. Uporablja se za reševanje dinamičnih problemov, ki izvirajo iz kompleksnih družbenih, vodstvenih, gospodarskih in ekoloških sistemov. Za te sisteme so značilni medsebojna odvisnost, medsebojna interakcija, povratni informacijski tokovi in krožna vzročnost. Za reševanje problemov se uporablja časovnozvezna simulacija modelov.

Problem dinamike kakovosti je kompleksen in časovno spremenljiv. Agregirani elementi socio-tehničnega sistema so medsebojno sklopljeni in nekatere vrednosti elementov so nelinearne. To upraviči uporabo metode sistemske dinamike za analizo in proučitev novih strategij.

4.1 Karakteristike orodja

Karakteristike razvoja, gradnje in uporabe orodja bodo razvrščene po tipologiji lastnosti porazdeljenih sistemov [54] (glej sliko 4.1).



Slika 4.1: Tipologija lastnosti porazdeljenih sistemov.

4.1.1 Transparentnost

Transparentnost je skrivanje kompleksnih podrobnosti sistema pred določenim objektom. Zanimala nas bo uporabniška transparentnost. To je skrivanje tehnoloških podrobnosti sistema, ki niso bistvene za opravljanje dela uporabnika. Skrivanje podrobnosti poenostavi uporabniško interakcijo s sistemom in zviša produktivnost in učinkovitost delovnega mesta.

Preizkuševalec opravlja dve nalogi za izpolnitev svojih delovnih obveznosti: (i) zasnova, razvoj in gradnja orodja in (ii) preizkušanje varnosti sistema. Oba vidika sta pomembna za produktivnost delovnega mesta. Zanimala nas bosta oba vidika uporabniške transparentnosti (glej tabelo 4.1 in 4.2). Boljša kot je podpora sistema, višja je stopnja transparentnosti sistema.

Tip	Vsebuje	Pomen
Transparentnost fizične gradnje značke	DA	Opis značke se nahaja v strojnem opisnem jeziku VHDL, Verilog ali SystemC. Realizacija značke na vezju FPGA pomeni, da razvijalec ne načrtuje na fizični ravni (raven tranzistorjev in geometrije vezja) in se izogne naslednjim postopkom: sinteza opisa logičnega vezja, identifikacija in uporaba gradnikov, povezovanje gradnikov, načrtovanje geometrije vezja z upoštevanjem časovnih omejitev in zakasnitev signalov, priprava mask za izdelavo vezja in proizvodnih postopkov integriranega vezja.
Transparentnost gradnje emulacijskega okolja	DA	Uporabniku ni potrebno poznati podrobnosti gradnje sistema MicroBlaze in operacijskega sistema Petalinux, saj obstaja podpora za avtomatizirano gradnjo.
Transparentnost opisa in gradnje komunikacijskega tunela	DA	Uporabniku ni potrebno znati realizirati strežnik komunikacijskega tunela čitalca. Opis je samodejen z uporabo orodja rpcgen, prav tako gradnja.

Tabela 4.1: Uporabniška transparentnost pri zasnovi, razvoju in gradnji orodja.

Tip	Vsebuje	Pomen
Transparentnost lokacije vira	DA	Uporabnik ne ve, kje se fizično nahaja emulacijsko okolje in ne čitalec. Oba vira sta oddaljena in se povezujeta preko interneta. Uporabniku ni potrebno vedeti naslova IP in ne vrat transportnega protokola TCP. Transparentnost je dosežena z uporabo storitev DNS in RPCBIND.
Transparentnost seljivosti vira	DA	Emulacijsko okolje ali čitalec je mogoče fizično preseliti, ne da bi uporabnik za to vedel. Orodje je mobilno.
Transparentnost replikacije vira	NE	Podpora za transparentno replikacijo značk, na primer na namen uravnoteženja bremen pri preizkušanju.
Transparentnost sočasnosti vira	NE	Dva različna preizkuševalca ne moreta sočasno uporabljati orodja, čitalca ali emulacijskega orodja.
Transparentnost paralele aktivnosti preizkušanja	NE	Orodje ne podpira (avtomatiziranega) paralelnega preizkušanja varnosti sistema.
Transparentnost trajnosti podatkov	NE	Stanje emulacijskega okolja značke se pri prekinitvi preizkušanja ne ohrani.
Transparentnost napake nezanesljive mrežne storitve	DA	Izgubljeni ukaz zaradi nezanesljive mrežne storitve bo zanesljivo znova prenešen po preteku časovne kontrole protokola TCP.

Tabela 4.2: Uporabniška transparentnost pri preizkušanju varnosti.

4.1.2 Prilagodljivost

Prilagodljivost je zmožnost spreminjanja, obdelovanja ali usklajevanja značilnosti sistema. Prilagodljivost je zelo zaželen lastnost orodja, ker je končna oblika tega orodja v splošnem nedorečena in odvisna od vsakega primera uporabe posebej. Višja stopnja prilagodljivosti pomeni nižjo ceno prilagajanja sistema.

Na prilagodljivost orodja vplivajo naslednji dejavniki:

- Zmožnost realizacije značke na vezju FPGA zmanjša čas načrtovanja in prihoda produkta na trg, saj se v primerjavi z ASIC izognemo načrtovanju na fizični ravni. Opis značke se nahaja v strojnem opisnem jeziku VHDL in tudi Verilog ter SystemC.
- Vezje FPGA je izdelano vnaprej, zato v primerjavi z ASIC odpade strošek proizvodnje prototipa značke.
- Uporaba programskega jezika C za realizacijo strežnika komunikacijskega tunela značke. Uporaba operacijskega sistema Linux in aplikacijskega programskega vmesnika POSIX za komunikacijo z emulacijskim okoljem značke.
- Specifikacija protokola za realizacijo strežnika komunikacijskega tunela čitalca je podana v jeziku RPCL. S pomočjo razpoložljivih orodij se opravi avtomatizirana transformacija za pridobitev izvirne kode strežnika za vrsto programskih jezikov: C, Java, Python, Perl itn. To poenostavi prilagoditev izvirne kode čitalca.
- Kontrola komunikacijskega mostu je ločena od uporabniškega grafičnega vmesnika (GUI), ki ga lahko nadomesti komponenta po meri, na primer komponenta za avtomatizirano preizkušanje varnosti.

4.1.3 Zanesljivost

Zanesljivost je lastnost sistema, da bo izvajal natanko tisto, kar hoče uporabnik, in to natanko takrat, ko se to od sistema zahteva. Za orodje so pomembne naslednje lastnosti izvajanja testov: atomarnost, konsistentnost, izolacija in trajnost (it angl. ACID).

Atomarnost testov ni zagotovljena zaradi majhnega števila pričakovanih napak in lažje izvedbe izboljšanja zmogljivosti orodja. Zato v primeru napake posledic ukaza ni mogoče preklicati in vrniti sistem v prejšnje stanje. Napaka je posredovana orodju, katerega naloga je rokovanje z napako in odprava napake s pomočjo dnevnika opravljenih testov.

Konsistentnost je zagotovljena, tako da vsi koraki posameznega testa vodijo sistem iz enega stabilnega stanja v drugo stabilno stanje.

- Dogodek prelitja vhodne krožne (sinhronizirane) vrste FIFO emulacijskega okolja ni mogoč (glej sliko 3.6). Če je vrsta polna, operacija pisanja javi napako (glej opis protokola komunikacijskega tunela značke 3.1 in čitalca 3.1. Ta napaka je

posredovana nazaj orodju kot rezultat klica oddaljene procedure. Naloga orodja je rokovanje s to napako.

- Izgubljeni ukaz zaradi nezanesljive omrežne storitve bo zanesljivo znova prenešen po preteku časovne kontrole protokola TCP.
- V primeru odpovedi strežnika komunikacijskega tunela značke (glej 3.2.3) ali čitalca (glej 3.2.5), je orodje obveščeno o napaki klica storitve. Naloga orodja je rokovanje z napako.

Isolacija testov je zagotovljena s sekvenčnim izvajanjem ukazov. Orodje ne podpira sočasnega večuporabniškega preizkušanja varnosti. Sekvenčno izvajanje je izvedeno z uporabo zanesljivih in blokirajočih klicev oddaljene procedure, ki prenesejo ukaz po komunikacijskem tunelu od odjemalca do strežnika. Strežnik komunikacijskega tunela značke in čitalca predstavimo z abstraktnim modelom strežnega sistema z eno vrsto in enim strežnim mestom. Sklepamo, da se pred prevzemom druge zahteve RPC prejšnja mora končati. Strežnik komunikacijskega tunela značke preda ukaz v vhodno (sinhronizirano) vrsto FIFO emulacijskega okolja (glej sliko 3.6). Krmilnik prevzame nov ukaz po tem, ko se izvajanje trenutnega konča in se rezultat operacije zapiše v izhodno vrsto. Z drugimi besedami povedano se vsi ukazi izvršijo v vrstnem redu, sočasnega izvajanja ukazov ni, zato je izolacija testov zagotovljena.

Trajnost testov ni zagotovljena. Naloga orodja je, da s pomočjo dnevnika vzpostavi zadnje stanje po izpadu sistema emulacijskega okolja.

4.1.4 Zmogljivost

Pri vrednotenju zmogljivosti orodja nas zanima maksimalno število testov v časovni enoti. Za izračun trajanja posameznega testa je potrebno poznati število klicov oddaljene procedure (glej diagram komunikacije sistema in uporabe orodja 3.21 ter opis serializacije sporočila čitalca v poglavju 3.2.6) in trajanje posameznega klica. Število klicev določa dolžina sporočila čitalca. Izračun časa trajanja enega testa predstavlja enačba 4.1, katere rešitev določata dva parametra: (i) zakasnitev čitalca in (ii) zakasnitev emulacijskega okolja. Čas računanja orodja za preizkušanje varnosti je zanemarljiv in se zanemari. Oba uporabljena parametra sta odvisna od dolžine sporočila čitalca in zakasnitev klica oddaljene procedure.

$$\begin{aligned}
 T_{en\ test}[ms] &= Zakasnitev\ \check{c}italca[ms] + Zakasnitev\ emulacijskega\ okolja[ms] \\
 &= Dolžina\ sporočila\ \check{c}italca[zlog] \times \frac{2\ klic}{zlog} \times Zakasnitev\ klica[\frac{ms}{klic}] \\
 &\quad + Dolžina\ sporočila\ \check{c}italca[zlog] \times 8 \frac{bit}{zlog} \times 2 \frac{klic}{bit} \\
 &\quad \times Zakasnitev\ klica[\frac{ms}{klic}]
 \end{aligned} \tag{4.1}$$

Za demonstracijski sistem je bila izbrana dolžina sporočila čitalca 1 zlog, povprečna izmerjena zakasnitev klica oddaljene procedure pa 11 ms. Čas izvajanja enega testa znaša približno 200 ms. Z orodjem je mogoče opraviti 18000 *testov/h*.

Prav tako nas zanima čas opisa in gradnje orodja. V kombinaciji s prejšnjo metriko izračunamo maksimalno število testov, ki ga ima preizkuševalec na voljo do naslednje iteracije preizkušanja. Dolžina opisa in gradnje orodja je odvisna od strukturnih sprememb značke (glej metodo zagotavljanja varnosti na sliki 3.29). Ocene za čas trajanja posameznih aktivnosti metode so podane spodaj in normalizirane na enega človeka. Režijski strošek učenja se zniža, če je predvidenih več iteracij preizkušanja.

- Čas trajanja definicije in gradnje operacijskega sistema Petalinux (P1) znaša 30 minut.
- Čas trajanja prilagoditve opisa značke za integracijo z orodjem (P2) je od 1 do 8 ur.
- Čas trajanja gradnje sistema MicroBlaze (P3, P4 in P5) znaša 1 uro.
- Čas trajanja opisa protokolnega analizatorja (P7) znaša od 1 do 8 ur.
- Čas trajanja prilagoditve čitalca in izvedbe vmesnika čitalca (P11) znaša 8 ur.

4.1.5 Razširljivost

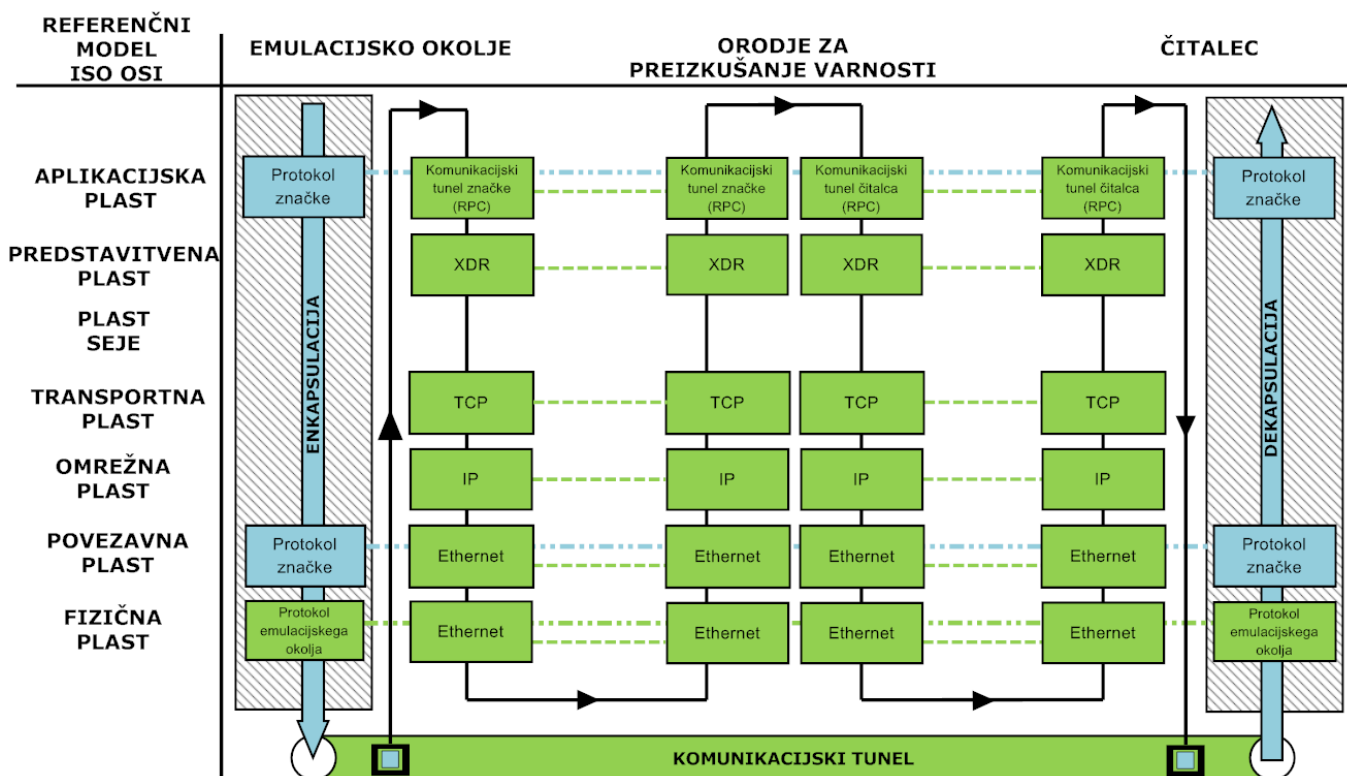
Razširljivost pomeni učinkovito prilagajanje orodja različnemu številu vozlišč in ohranjanje sprejemljive odzivnosti in primerne kakovosti storitev.

Trenutna izvedba orodja ni razširljiva. Višja stopnja razširljivosti je dosežena le s paralelno uporabo več orodij za preizkušanje naenkrat.

4.2 Omrežna arhitektura orodja

Na sliki 4.2 je prikazan omrežni arhitekturni model orodja, katerega protokoli so razvrščeni vertikalno glede na referenčni model ISO OSI in horizontalno glede na tip vozlišča. Prikazan je pretok podatkov po informacijsko-komunikacijskem skladu in logično povezovanje posameznih plasti.

Povezovanje značke in čitalca poteka posredno preko komunikacijskega tunela. Emulacijsko okolje in orodje za preizkušanje varnosti se povezujeta s protokolom komunikacijski tunel značke, ki je realiziran s protokolom RPC. Protokolni konverter orodja za preizkušanje varnosti poskrbi za transformacijo podatka in posredovanje sporočila čitalcu z uporabo protokola komunikacijski tunel čitalca, ki je prav tako realiziran s protokolom RPC. Z orodjem za zagotavljanje varnosti nadzorujemo izvajanje protokolnega konverterja za nadzor nad komunikacijskim kanalom značke in čitalca ter nad logičnim izvajanjem značke.



Slika 4.2: Omrežna arhitektura orodja.

4.3 Omejitve orodja

Orodje ima naslednje omejitve:

- Uporaba orodja za preizkušanje temelji na pravilno prepoznanih grožnjah, saj bo preizkuševalec poskušal izvesti napad na značko in zlorabiti sistem podobno kot napadalec. To pomeni, da je orodje namenjeno varnostnemu strokovanjaku za bolj produktivno izvajanje nalog.
- Preizkušanje ranljivosti ni izčrpno[†]. Uspešnost metode temelji na pametno izbranih testih glede na prepoznane grožnje. Z uporabo orodja se potrdi prisotnost ranljivosti značk, a odsotnost ranljivosti ni mogoče potrditi.
- Izvedba orodja podpira preizkušanje varnosti sinhronih digitalnih vezij, asinhrona digitalna vezja niso podprta.
- Originalna fizična komunikacijska plast značke je v orodju neizvedljiva, če je njena logika asinhrona. Zato jo nadomesti posebna fizična plast emulacijskega okolja. To pomeni, da morebitne ranljivosti fizične plasti ni mogoče odkriti. Primer ranljivosti fizične plasti je podan v [28]. Podobno ni mogoče preizkusiti protiukrepov na fizični

[†]Trenutno ne obstaja primer orodja za izčrpno preizkušanje lastnosti kompleksnega in heterogenega sistema. Izčrpno so preizkušene le skrbno izbrane lastnosti kritične komponente sistema.

plasti za učinkovitost in pravilnost. Primer protiukrepa na fizični plasti je podan v [55].

- Vezja FPGA imajo v primerjavi z ASIC večjo površino in večjo porabo energije, manj optimalno postavitev gradnikov in manj optimalno izvedbo povezav med gradniki. Zaradi prilagodljivosti se v povezovalni mreži nahajajo aktivni elementi, ki vnašajo še dodatne zakasnitve signalov.
- Strošek nakupa strojne in programske opreme za izdelavo orodja v letu 2011 znaša približno 2000 EUR.

4.4 Dinamika kakovosti sistema

Deležniki s poslovnimi cilji si na začetku uporabe omrežnih sistemov RFID ustvarijo začetno predstavo groženj in imajo začetno zavedanje o varnosti sistema. Začetno zavedanje oblikujejo različni dejavniki: pretekle izkušnje s podobno tehnologijo, ugled združbe in prepričljivost argumenta o varnosti sistema, ki ga podajo razvijalci sistema in združba, ki ponuja storitve s pomočjo sistema.

Zgodovina izkoriščanja ranljivosti sistema RFID priča, da razvijalci sistemov in združbe ne dajo zadostnega pomena zagotavljanju varnosti sistema. V to jih ne silijo niti deležniki, saj jih o varnosti prepričajo že osebna stališča razvijalcev sistema in tržnikov storitev. Razvijalci in združbe vidijo dodano varnost kot strošek, ki draži razvoj in postavitev sistema ter odlaša čas prihoda storitve na trg. Le-to vodi v oblikovanje preoptimistične začetne ocene tveganja deležnikov o varnosti sistema. Sistem je izpostavljen in po preteku določenega obdobja hekerji razkrijejo ranljivosti. Hekerji te ranljivosti tudi izkoriščajo. Deležniki po določenem času zaznajo napad na sistem, saj pri tem izgubijo dragocena sredstva. Po tem spoznanju začetno oceno tveganja prilagodijo, saj se začnejo zavedati pomanjkljivosti sistema [17]. Prilagoditev ocene tveganja in prilagoditev zavedanja deležnikov o varnosti sistema zniža zadovoljstvo deležnikov in ostale karakteristike kakovosti sistema po ISO/IEC 9126-1 [16], ki so v medsebojni odvisnosti.

Združbe upravljajo kakovost storitve in ukrepajo v primeru spremembe ciljne ravni kakovosti. Ranljivost izpostavlja sistem različnim grožnjam in vpliva na kakovost storitve. Združbe spremljajo in pregledujejo tveganja poslovnega sistema. Prisotnost ranljivosti v najboljšem primeru odkrijejo, preden je izkoriščena, v najslabšem pa prepozno, ko so deležniki nezadovoljni in je škoda že povzročena.

Neodvisno od odzivnega časa združbe ukrepajo in poskušajo odpraviti ranljivost s pomočjo razvijalcev sistema. To storijo v procesu obvladovanja tveganja, ki je sestavljen iz več aktivnosti. Najprej prepoznajo tveganja. Ta tveganja ocenijo, nato razvijejo načrt za obravnavo tveganja. Potem ko tveganja sprejmejo, sledi vpeljava načrta za obravnavo tveganja. Vsaka faza obvladovanja tveganja traja določen čas, zato je za minimizacijo škode odločilno, da je proces načrtovanja in vpeljave novega sistema za upravljanje varovanja informacij hiter in učinkovit.

4.4.1 Model

Model, ki upošteva strukturo zgornjega problema je predstavljen na sliki 4.3[†] in razširi model vzroka in posledic, ki smo ga opisali v pregledu področja (glej 2.4). Pripadajoči matematični opis modela se nahaja v privesku G.

Na obnašanje modela vplivajo tri zanke.

1. Zanka R1 je povratna zanka produktivnosti sistema. Večje zadovoljstvo deležnikov pozitivno vpliva na produktivnost in obratno.
2. Zanka R2 je povratna zanka učinkovitosti sistema. Večje zadovoljstvo deležnikov pozitivno vpliva na učinkovitost in obratno.
3. Zanka B1 je povratna zanka obvladovanja tveganja, ki poskuša ohranjati sistem v stabilnem stanju na sprejemljivi ravni tveganja. Če se varnost zmanjša oziroma tveganje zviša, nastane razlika od sprejemljive ravni tveganja in vpeljejo se protiukrepi, ki zvišajo varnost na želeno raven. Zanka B1 je realizacija procesa obvladovanja tveganja ISO/IEC 27005:2008 [30], ki je prikazan na sliki 2.5.

Trenutno zadovoljstvo deležnikov je uravnotežena vsota prispevkov zavedanja deležnikov o varnosti sistema, produktivnosti in učinkovitosti sistema. Trenutno zadovoljstvo izoblikuje dolgoročno zadovoljstvo deležnikov po preteku določenega časa. Rečemo, da ima dolgoročno zadovoljstvo inercijo.

Povedano z drugimi besedami in v strnjeni obliki, problem obvladovanja kakovosti sistema z vidika varnosti je problem učinkovitega obvladovanja tveganja pri varovanju informacij. Pri tem sprejemljivo raven tveganja določijo deležniki. Problematična dinamika kakovosti z vidika varnosti izvira iz medsebojne povezanosti karakteristik kakovosti, zakasnenih vplivov med posameznimi karakteristikami in omejenimi možnostmi za učinkovito obvladovanje tveganja.

4.4.2 Simulacije

Pri simulaciji modela nas bo zanimala dinamika kakovosti sistema glede na kakovost obvladovanja tveganja okolij omrežnih sistemov RFID. Pokazali bomo, da model generira referenčno obnašanje, ki je podano z zgornjim kvalitativnim opisom.

Za ta namen bomo ustvarili različne scenarije. Vsak scenarij bo imel drugačne razloge za spremembo varnosti sistema. Glede na čas nastanka ranljivosti bodo scenariji različno poimenovani. Ločili bomo nastanek ranljivosti pri razvoju in postavitvi sistema in nastanek ranljivosti pri zasnovi, razvoju in postavitvi sistema. Za vsak scenarij bomo pregledali tri možne izide.

[†]Na diagramu SD G.1 so predstavljeni le najbolj bistveni elementi. Popolna struktura modela se nahaja na sliki G.1 v privesku G.

1. Slab izid bo predstavljal referenčno obnašanje. To je izid, pri katerem hekerji ranljivost razkrijejo pred odgovornimi, zato je povzročena škoda največja. Deležniki se najbolj bojijo ravno tega izida.
2. Srednje dober izid bo predstavljal izboljšano reakcijo odgovornih na novo ranljivost. Čas obravnave in sprejetja tveganja odgovornih ter na drugi strani razkritja ranljivosti hekerjev bosta enaka. Zaradi končne kapacitete delovne moči bodo protiukrepi vpeljeni v določenem času in to časovno okno bodo hekerji izkoriščali ranljivosti. Škoda bo za raličo od prejšnjega primera manjša.
3. Dober izid bo predstavljal uporabo orodja za zagotavljanje varnosti za okolja omrežnih sistemov RFID v procesu obvladovanja tveganja. Pri tem bomo privzeli, da se izvor ranljivosti sistema nahaja v znački. Tveganja bodo pravočasno obravnavana in sprejeta in protiukrepi bodo vpeljeni še preden bo povzročena večja škoda. Škoda bo najmanjša ali pa je celo ne bo.

Različni izidi scenarijev bodo posledica različne kakovosti obvladovanja tveganja. Vzvodi sprememb kakovosti obvladovanja tveganja so na diagramu označeni odebeljeno (glej sliko 4.3). Uporaba orodja in metode za zagotavljanje varnosti za okolja omrežnih sistemov RFID ima v primerjavi s pristopom “ad hoc” naslednje prednosti:

- Uporaba orodja in metode zmanjša čas prepoznavne tveganja in čas obravnave in sprejetja tveganja, pri čemer se določijo primerni in učinkoviti protiukrepi za zmanjšanje tveganja na sprejemljivo raven.
- Če se orodja in metoda uporabljata pri zasnovi in razvoju sistema, je začetna varnost sistema pri postavitvi v povprečju boljša.
- Metoda in orodje predstavljata strukturiran pristop k obvladovanju tveganja za okolja omrežnih sistemov RFID. Uporaba strukturiranega pristopa namesto pristopa “ad hoc” predstavlja boljše zagotovilo deležnikom, da so njihove varnostne zahteve zagotovljene, saj so bile obširneje preverjene. Posledično se zmanjša čas za izboljšanje zavedanja deležnikov o varnosti sistema.

Pri opazovanju in primerjavi izidov scenarija bomo opazovali rdeče označene spremenljivke modela (glej sliko 4.3). Produktivnost sistema, učinkovitost sistema in zadovoljstvo sistema bodo zakasneni pokazatelji kakovosti obvladovanja tveganja, medtem ko primerjava izkoriščanja ranljivosti in posameznih faz obvladovanja tveganja vodilni pokazatelji obvladovanja tveganja.

Privzete začetne vrednosti simulacije so navedene v tabeli 4.3. Celoten seznam vrednosti se nahaja v matematičnem opisu modela (glej privesek G).

Spremenljivka	Vrednost	Opis
Čas trajanja simulacije	52 mesecev	
Časovni korak integracije	22 ur	Eulerjeva integracijska metoda.
Začetna produktivnost sistema	50 %	Povprečna vrednost s priložnostjo za izboljšanje.
Začetna učinkovitost sistema	50 %	Povprečna vrednost s priložnostjo za izboljšanje.
Začetno zadovoljstvo deležnikov	100 %	Idealna vrednost, temelji na tehnološkem navdušenju deležnikov.
Začetna varnost sistema	100 %	Idealna vrednost.
Začetno zavedanje deležnikov o varnosti sistema	100 %	Idealna vrednost, če je začetna varnost sistema enaka 100 %.
Čas izboljšanja zavedanja deležnikov o varnosti sistema	12 mesecev	Povprečna vrednost.
Normirani čas razkritja ranljivosti	6 mesecev	Arbitrarno določena vrednost.
Normirani čas prepoznavanja tveganja	4 mesece	Slabo, saj je ta čas skupaj s časom obravnave in sprejetja tveganja ter vpeljave protiukrepov večji od časa razkritja ranljivosti in izkoriščanja ranljivosti.
Normiran čas obravnave in sprejetja tveganja	4 mesece	Slabo, saj je ta čas skupaj s časom prepoznave tveganja in vpeljave protiukrepov večji od časa razkritja ranljivosti in izkoriščanja ranljivosti.
Čas vpeljave protiukrepov	1 mesec	Slabo, saj je ta čas skupaj s časom prepoznave tveganja, časom obravnave in sprejetja tveganja večji od časa razkritja ranljivosti in izkoriščanja ranljivosti.

Tabela 4.3: Privzeti začetni pogoji modela.

4.4.3 Nova ranljivost pri aktivni uporabi sistema

V prvem scenariju nas bo zanimala dinamika kakovosti sistema, ko ranljivost nastane pri aktivni uporabi sistema.

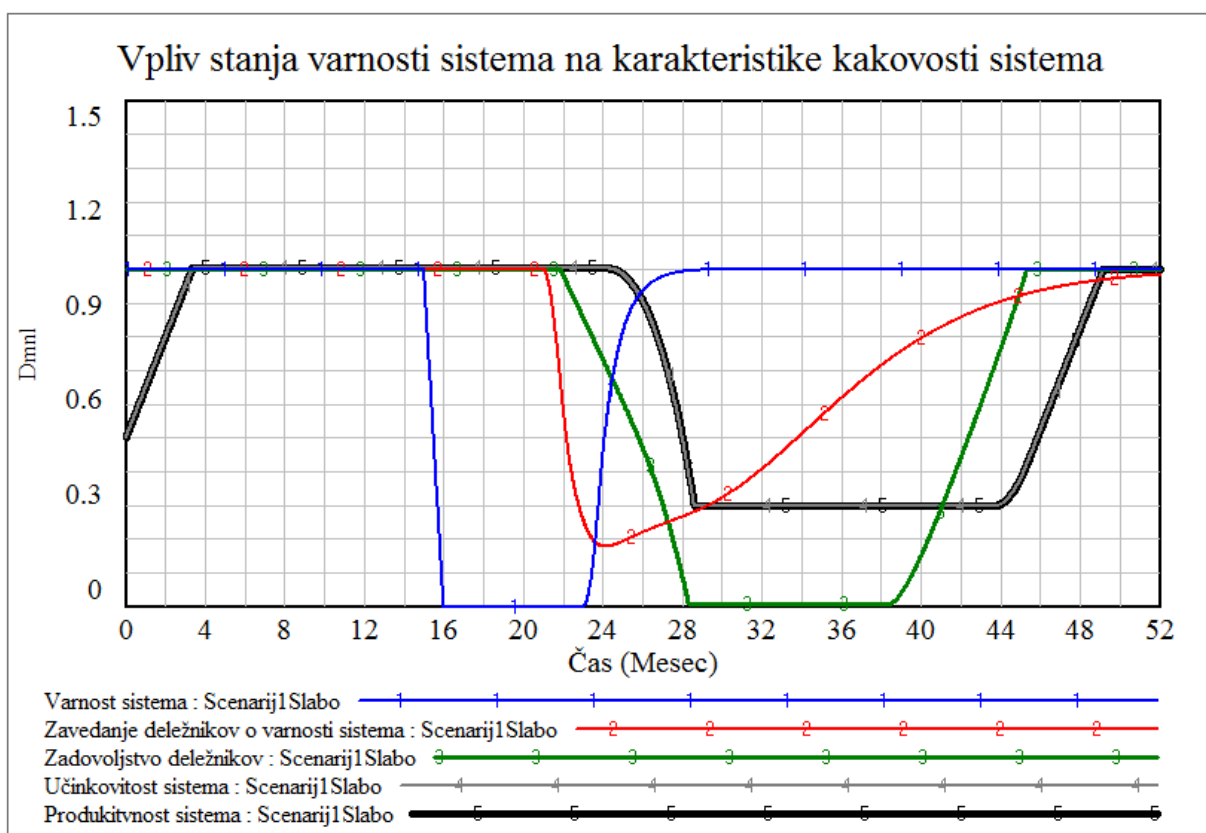
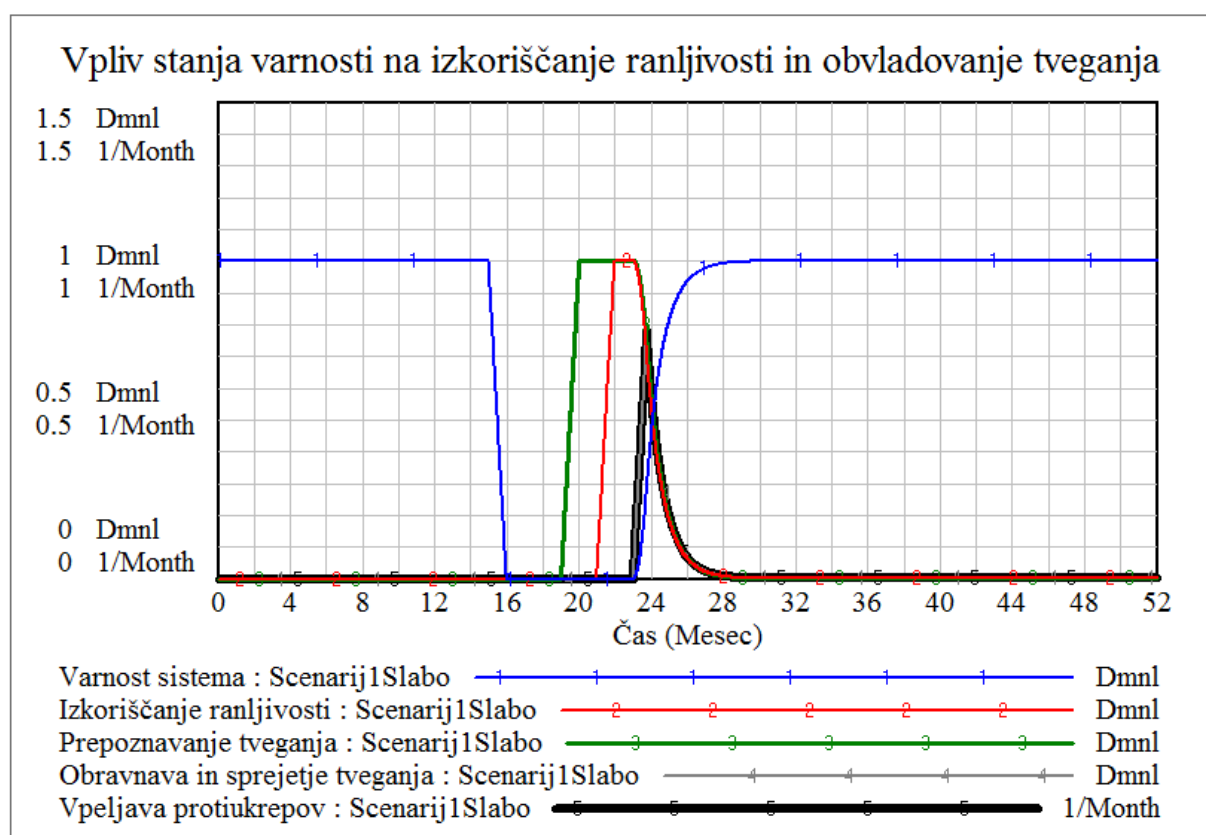
Začetni pogoji so podani v tabeli 4.3. Začetno zavedanje varnosti sistema in realna varnost sistema se ne razlikujeta. Prepričanje deležnikov o začetni varnosti sistema je pravilno. Zanima nas dinamika karakteristik kakovosti sistema, ki jo sproži varnostni incident.

Slaba kakovost obvladovanja tveganja

Pri slabi kakovosti obvladovanja tveganja so uporabljeni privzeti začetni pogoji modela (glej tabelo 4.3). Diagrama na sliki 4.4 prikazujeta obnašanje modela.

Na zgornjem diagramu je prikazano stanje varnosti (zg. diag, črta 1), ki je na začetku popolno 100%. Po nastanku ranljivosti se stanje varnosti takoj zmanjša na 0 %. Hekerji začnejo izkoriščati ranljivosti po preteku časa razkritja ranljivosti (zg. diag., črta 4). Posledica izkoriščanja je prilagoditev zavedanja deležnikov o varnosti sistema, ki kasni glede na čas nastanka ranljivosti (sp. diag., črta 1 in črta 2). Odziv prve faze obvladovanja tveganja, tj. prepoznavanje tveganja, prehiteva izkoriščanje ranljivosti (zg. diag., črta 2 in 3), vendar so protiukrepi vpeljani prepozno zaradi slabe učinkovitosti obvladovanja tveganja (zg. diag, črta 4 in 5). Okno izkoriščanja ranljivosti je široko (zg. diag., vrzel črte 1), zato je povzročena škoda velika (zg. diag., površina pod črto 2). Izkoriščanje ranljivosti se ustavi, ko so vpeljani vsi protiukrepi (zg. diag., črta 5).

Deležniki daljši čas niso prepričani v pravilnost in učinkovitost protiukrepov zaradi pristopa "ad hoc" obvladovanja tveganja. Posledica tega je, da se varnost sistema in zavedanje deležnikov o varnosti sistema daljši čas razlikuje (sp. diag., črta 1 in 2 od 24. meseca naprej). To povzroči dolgotrajno nezadovoljstvo (sp. diag., črta 3), ki ima kaskadni vpliv tudi na produktivnost in učinkovitost sistema (sp.diag, črta 4 in 5). Zadnji dve karakteristiki sta odgovorni za negativne ekonomske posledice, ki trajajo 24 mesecev.



Slika 4.4: Scenarij “Nova ranljivost pri aktivni uporabi sistema” – Slaba kakovost obvladovanja tveganja.

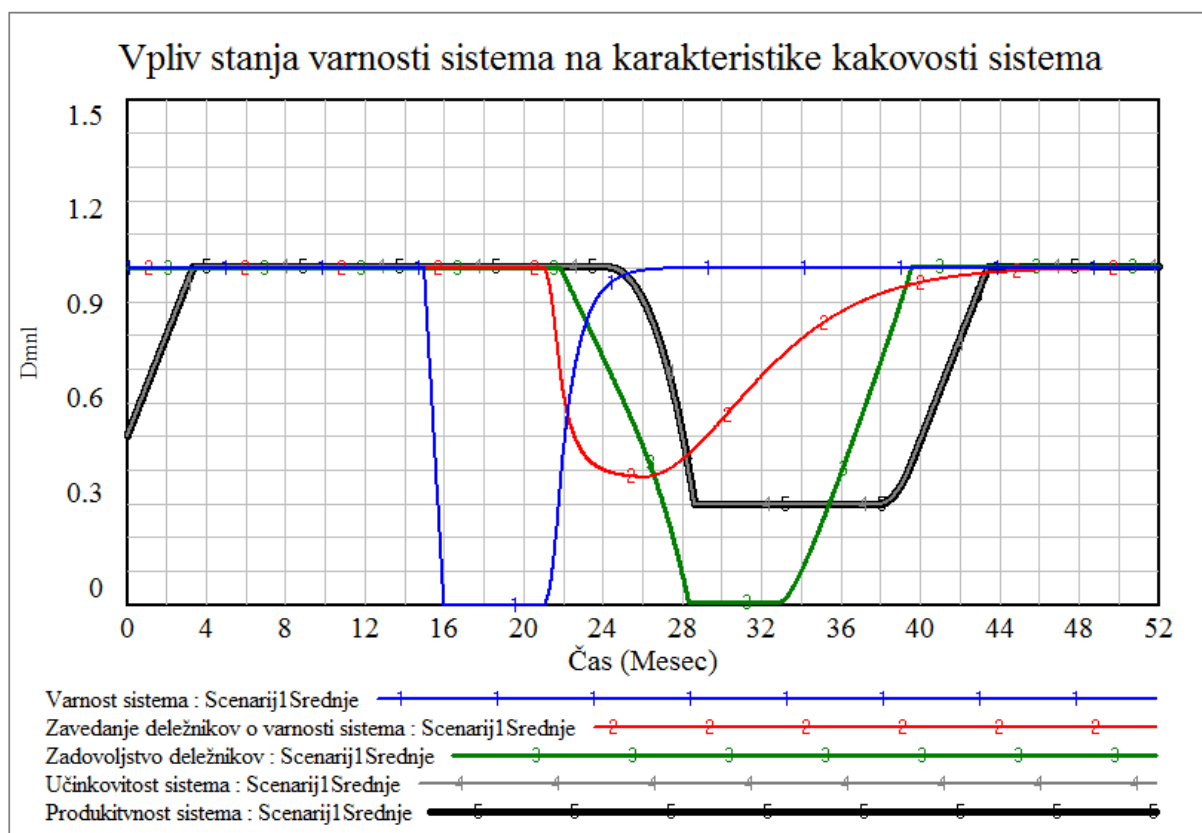
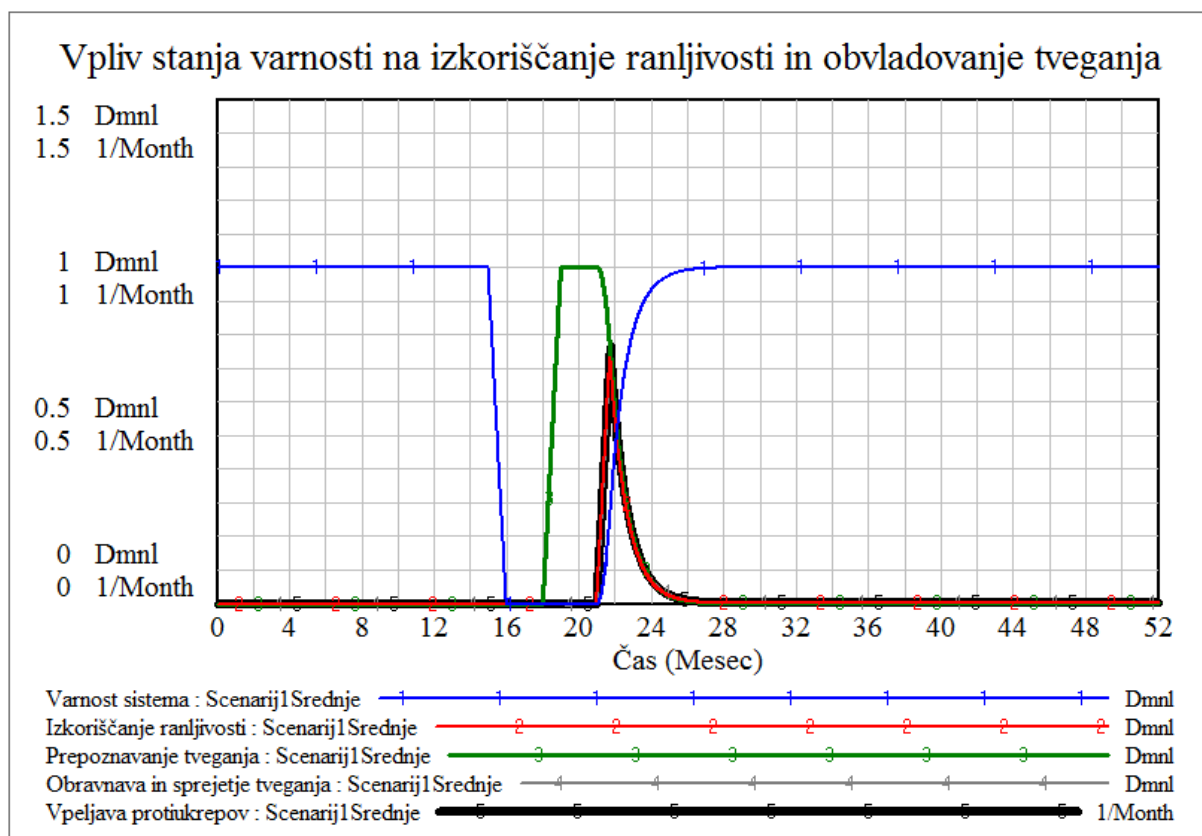
Srednja kakovost obvladovanja tveganja

Pri srednji kakovosti obvladovanja tveganja so spremenjeni privzeti začetni pogoji modela. Na spremembo začetnih pogojev vpliva uporaba orodja in metode pri obvladovanju tveganja in za zagotavljanje varnosti. V spodnji tabeli se nahajajo spremenjene vrednosti z dodanim komentarjem vzroka sprememb.

Spremenljivka	Vrednost	Razlog za spremembo
Čas izboljšanja zavedanja deležnikov o varnosti sistema	12 → 9 mesecev	Uporaba orodja in metode pripomore k bolj prepričljivemu zagotavljanju varnostnih funkcionalnosti. Posledično se zmanjša čas za izboljšanje zavedanja deležnikov o varnosti sistema, saj več deležnikov verjame v varnost sistema.
Čas prepoznavanja tveganja	5 → 3 mesecev	Uporaba strukturiranega pristopa zmanjša čas prepoznavanja tveganja.
Čas obravnave in sprejetja tveganja	5 → 3 mesecev	Uporaba strukturiranega pristopa zmanjša čas obravnave in sprejetja tveganja.

Tabela 4.4: Sprememba začetnih vrednosti simulacije za scenarij “Nova ranljivost pri aktivni uporabi sistema” – Srednja kakovost obvladovanja tveganja.

Diagrama na sliki 4.5 prikazujeta obnašanje modela. Dinamika izkoriščanja ranljivosti in vpeljava protiukrepov sovpadata (zg. diag., črta 2 in 5). Okno izkoriščanja je ožje v primerjavi s prejšnjim izidom (zg. diag., vrzel črte 1), zato je škoda manjša (zg. diag., površina pod črto 2). Posledično je manjši padec zavedanja deležnikov o varnosti sistema (sp. diag., črta 2), kar ima pozitiven učinek na zadovoljstvo deležnikov (sp. diag., črta 3), učinkovitost sistema (sp. diag., črta 4) in produktivnost sistema (sp. diag., črta 5). Padec učinkovitosti in produktivnosti imata negativne ekonomske posledice za obdobje 18 mesecev. V primerjavi s prejšnjim izidom je to 6 mesecev manj.



Slika 4.5: Scenarij “Naknadno odkrita ranljivost” – Srednja kakovost obvladovanja tveganja.

Dobra kakovost obvladovanja tveganja

Pri dobri kakovosti obvladovanja tveganja so spremenjeni privzeti začetni pogoji modela. Metoda in orodje sta izkoriščena na najboljši način.

Spremenljivka	Vrednost	Razlog za spremembo
Čas izboljšanja zavedanja deležnikov o varnosti sistema	12 → 6 mesecev	Uporaba orodja in metode pripomore k bolj prepričljivemu zagotavljanju varnostnih funkcionalnosti. Posledično se zmanjša čas za izboljšanje zavedanja deležnikov o varnosti sistema, saj več deležnikov verjame v varnost sistema.
Čas prepoznavanja tveganja	5 → 2 mesecev	Uporaba strukturiranega pristopa zmanjša čas prepoznavanja tveganja.
Čas obravnave in sprejetja tveganja	5 → 2 mesecev	Uporaba strukturiranega pristopa zmanjša čas obravnave in sprejetja tveganja.

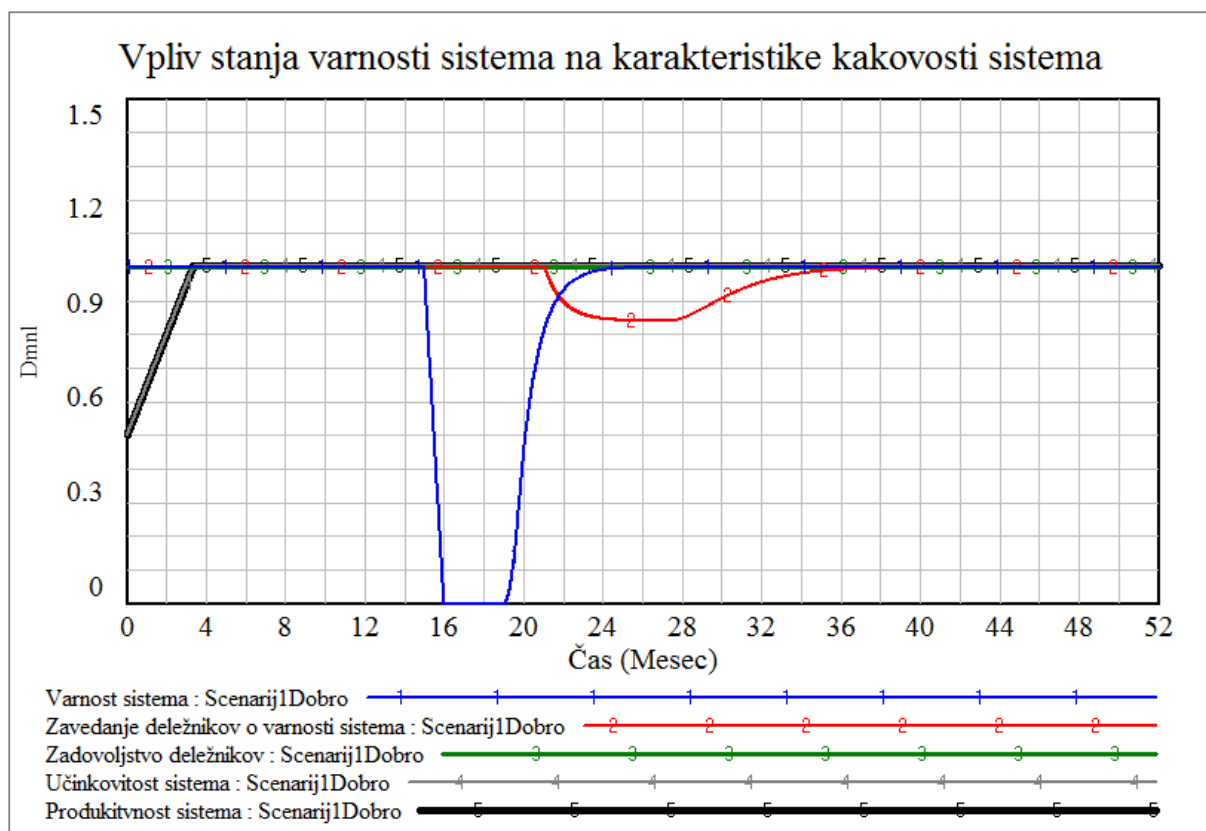
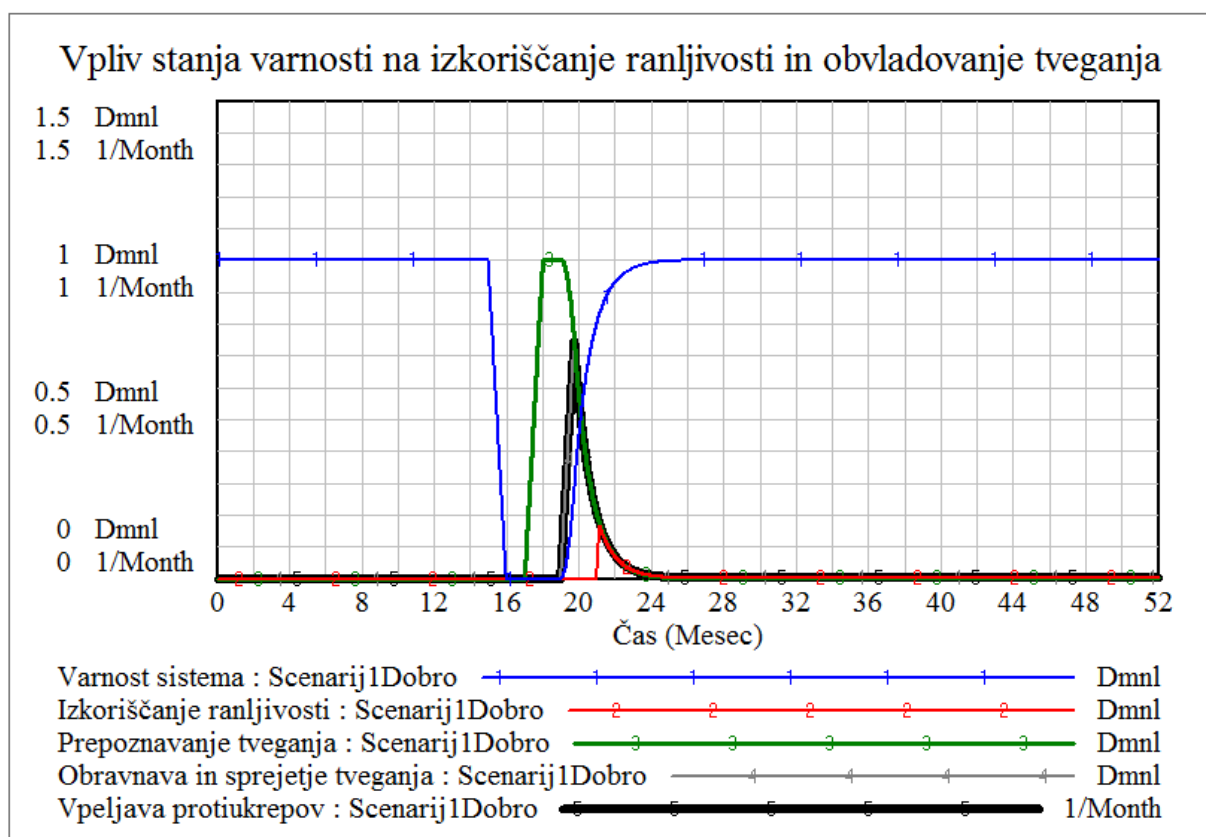
Tabela 4.5: Začetni pogoji modela za scenarij “Nova ranljivost pri aktivni uporabi sistema” – Dobra kakovost obvladovanja tveganja.

Diagrama na sliki 4.6 prikazujeta obnašanje modela. Vse faze obvladovanja tveganja preHITEVajo incidente (zg. diag., črta 2): (i) prepoznavanje tveganja (zg. diag., črta 3), (ii) obravnava in sprejetje tveganja (zg. diag., črta 4) ter (iii) vpeljava protiukrepov (zg. diag., črta 5). Okno izkoriščanja ranljivosti je najmanjše (zg. diag., vrzel črte 1). Škoda je minimalna (zg. diag., površina pod črto 2), saj je večina protiukrepov vpeljana pravočasno (zg. diag., črta 5). Majhna škoda povzroči zanemarljiv padec zavedanja deležnikov o varnosti sistema (sp. diag., črta 2) in nima učinka na zadovoljstvo deležnikov (sp. diag., črta 3), učinkovitost (sp. diag., črta 4) in produktivnost sistema (sp. diag., črta 5). Zaradi dobrega obvladovanja tveganja ni negativnih ekonomskih posledic.

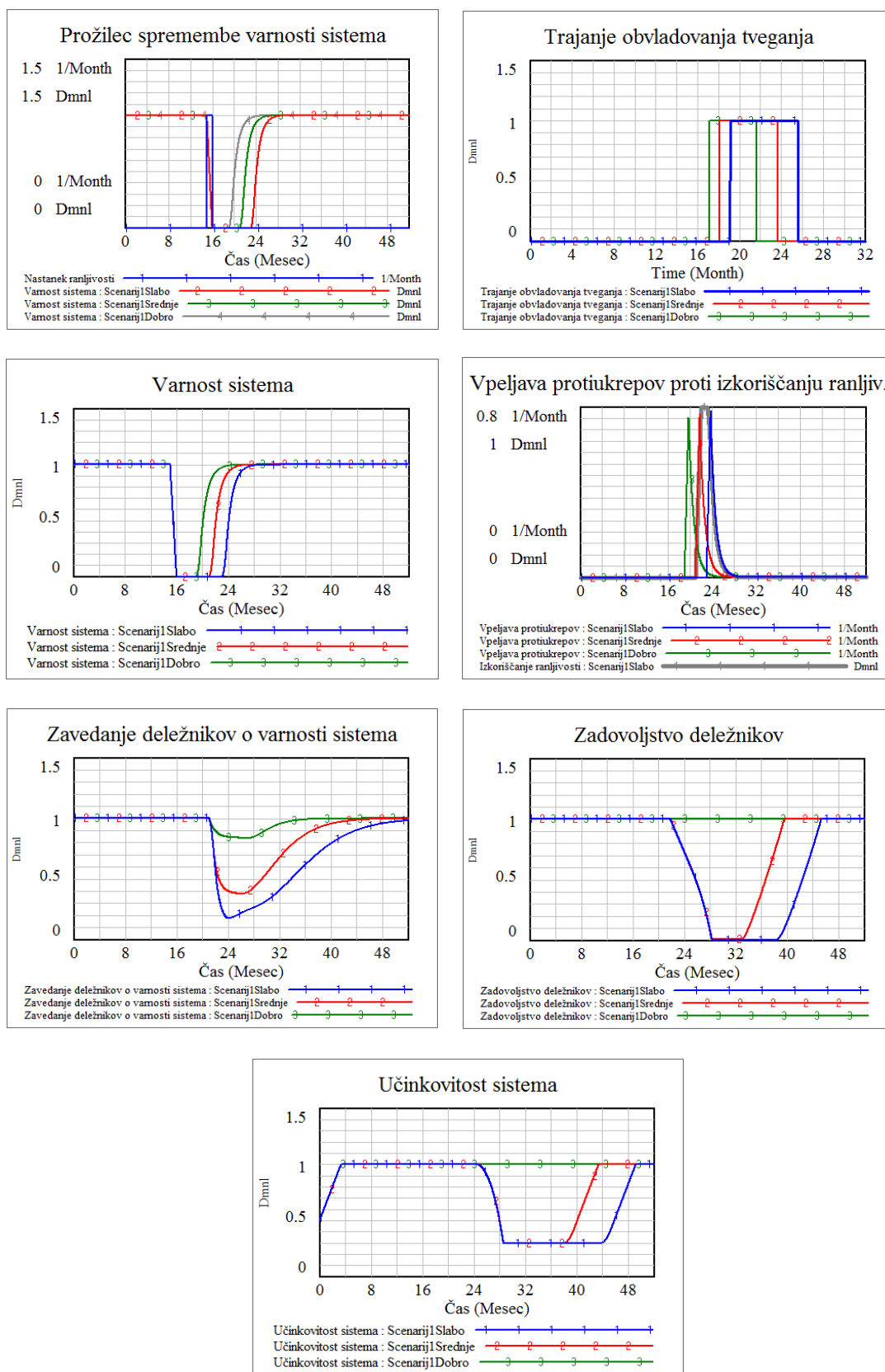
Primerjava izidov

Na sliki 4.7 so zbrane primerjave vseh treh izidov scenarija “Nova ranljivost pri aktivni uporabi sistema”. Prožilec je v vseh izidih scenarija nastanek nove ranljivosti pri aktivni uporabi sistema.

Vsi izidi scenarijev uporabljajo proaktivni pristop obvladovanja tveganja, kjer so tveganja prepoznana pred dogodkom incidenta. V najboljšem primeru so protiukrepi vpeljani pred razkritjem ranljivosti hekerjev. Izkoriščanje ranljivosti in s tem povezana škoda je odvisna od učinkovitosti obvladovanja tveganja. V realnosti združbe navadno uporabljajo reaktivni pristop obvladovanja tveganja. Zaradi tega bi bila škoda še večja od najslabšega izida. Pri reaktivnem pristopu bi faza prepoznavanja tveganja zaostajala glede na prepoznavanje tveganja. Ekonomska škoda, ki bi jo utrpeli deležniki, bi bila gromozanska.



Slika 4.6: Scenarij “Nova ranljivost pri aktivni uporabi sistema” – Dobra kakovost obvladovanja tveganja.



Slika 4.7: Primerjava obnašanja spremenljivk sistema treh izidov scenarija “Nova ranljivost pri aktivni uporabi sistema”.

4.4.4 Nabava ranljivega sistema

V drugem scenariju nas bo zanimala dinamika kakovosti sistema, katere prožilec je ranljivost, ki je prisotna še pred postavitvijo sistema. Le-ta je lahko posledica nekakovostne zasnove in razvoja sistema ali pa slabe presoje vodilnih pri nakupu sistema. Vzroka za slabo presojo združb sta naslednja.

- Združbe imajo nezadostne in neučinkovite zahteve o varnostni funkcionalnosti pri nabavi sistema zaradi nepoznavanja varnostnih zahtev poslovnega sistema.
- Združbe imajo nezadostne zahteve o stopnji zanesljivosti ocene o varnostnih funkcionalnostih sistema zaradi nepoznavanja stopnje tveganja poslovnega sistema.

V drugem scenariju se začetno zavedanje deležnikov o varnosti sistema in dejanska začetna varnost sistema razlikujeta. Deležniki in združba imajo preoptimistično prepričanje o varnosti sistema. Za ta scenarij so spremenjeni nekateri privzeti začetni pogoji modela (glej tabelo 4.3). Spremembe so navedene v tabeli 4.6.

Spremenljivka	Vrednost	Opis
Začetna varnost sistema	30 %	Slaba začetna varnost.
Začetno zavedanje deležnikov o varnosti sistema	100 %	Slabo, saj se razlikuje od dejanske začetne varnosti sistema.

Tabela 4.6: Privzeti začetni pogoji modela za scenarij “Nabava ranljivega sistema”.

Slaba kakovost obvladovanja tveganja

Pri slabi kakovosti obvladovanja tveganja so uporabljeni privzeti začetni pogoji tabele 4.6. Diagrama na sliki 4.8 prikazujeta obnašanje modela.

Na zgornjem diagramu je prikazano stanje varnosti (zg. diag., črta 1), katerega začetna vrednost je slaba in znaša 30 %. Tako slaba začetna vrednost varnosti sistema izhaja iz nekakovostne zasnove in razvoja sistema ali pa je posledica slabe presoje vodilnih združb pri nakupu sistema. Hekerji začnejo izkoriščati ranljivost po preteku časa razkritja ranljivosti (zg. diag., črta 4). Posledica izkoriščanja je škoda, ki jo občutijo deležniki. Sledi prilagoditev zavedanja deležnikov o varnosti sistema (sp. diag., črta 2). Odziv prve faze obvladovanja tveganja, tj. prepoznavanje tveganja, prehiteva izkoriščanje ranljivosti (zg. diag., črta 2 in 3). Kljub temu nastane škoda zaradi izkoriščanja ranljivosti, saj so protiukrepi vpeljani prepozno zaradi slabe učinkovitosti procesa obvladovanja tveganja (zg. diag., črta 4 in 5). Okno izkoriščanja ranljivosti je široko (zg. diag., vrzel črte 1), zato je povzročena škoda velika (zg. diag., površina pod črto 2). Izkoriščanje se ustavi, ko so vpeljani vsi protiukrepi (zg. diag., črta 5).

Deležniki daljši čas niso prepričani v pravilnost in učinkovitost protiukrepov zaradi pristopa “ad hoc” obvladovanja tveganja. Posledica tega je, da se varnost sistema in zavedanje deležnikov o varnosti sistema daljši čas razlikujeta (sp. diag., črta 1 in 2 od 14.

meseca naprej). To povzroči dolgotrajno nezadovoljstvo (sp. diag., črta 3), ki ima kaskadni vpliv tudi na produktivnost in učinkovitost sistema (sp. diag., črta 4 in 5). Negativne ekonomske posledice trajajo za obdobje padca zadnjih dveh karakteristik 24 mesecev.

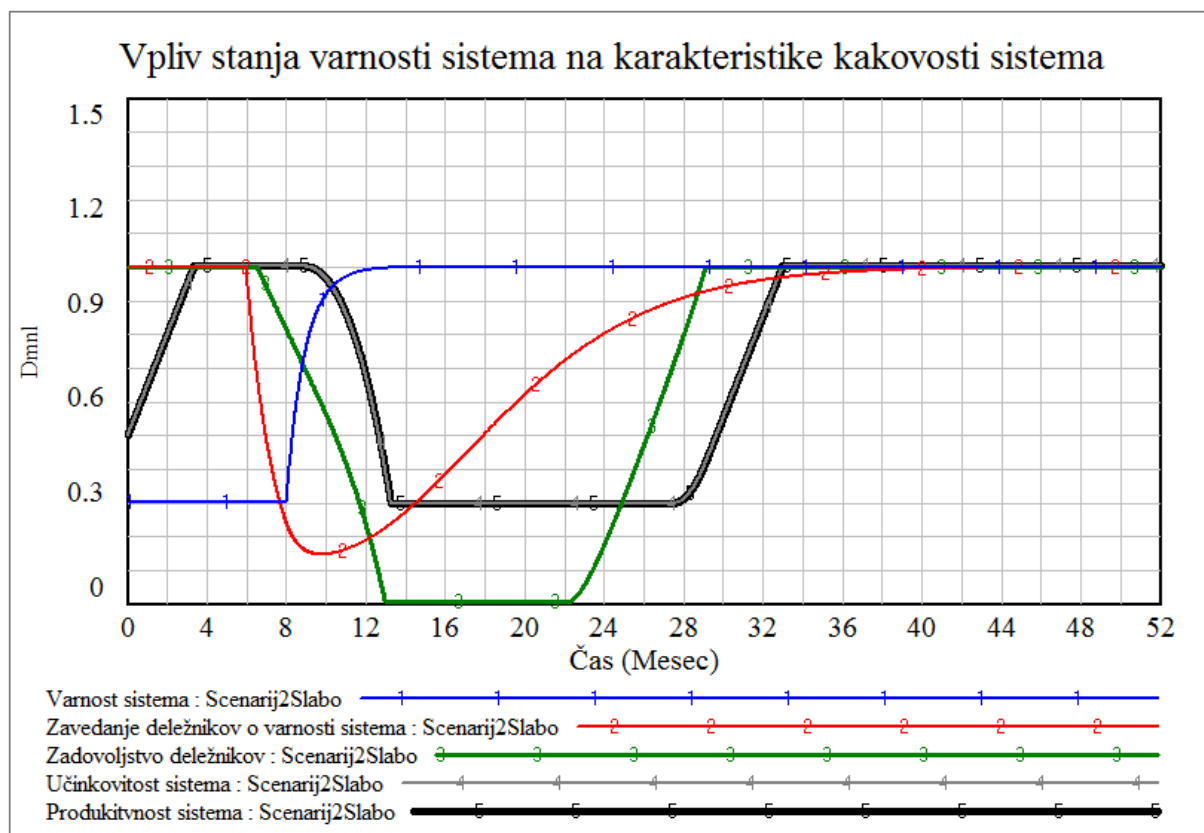
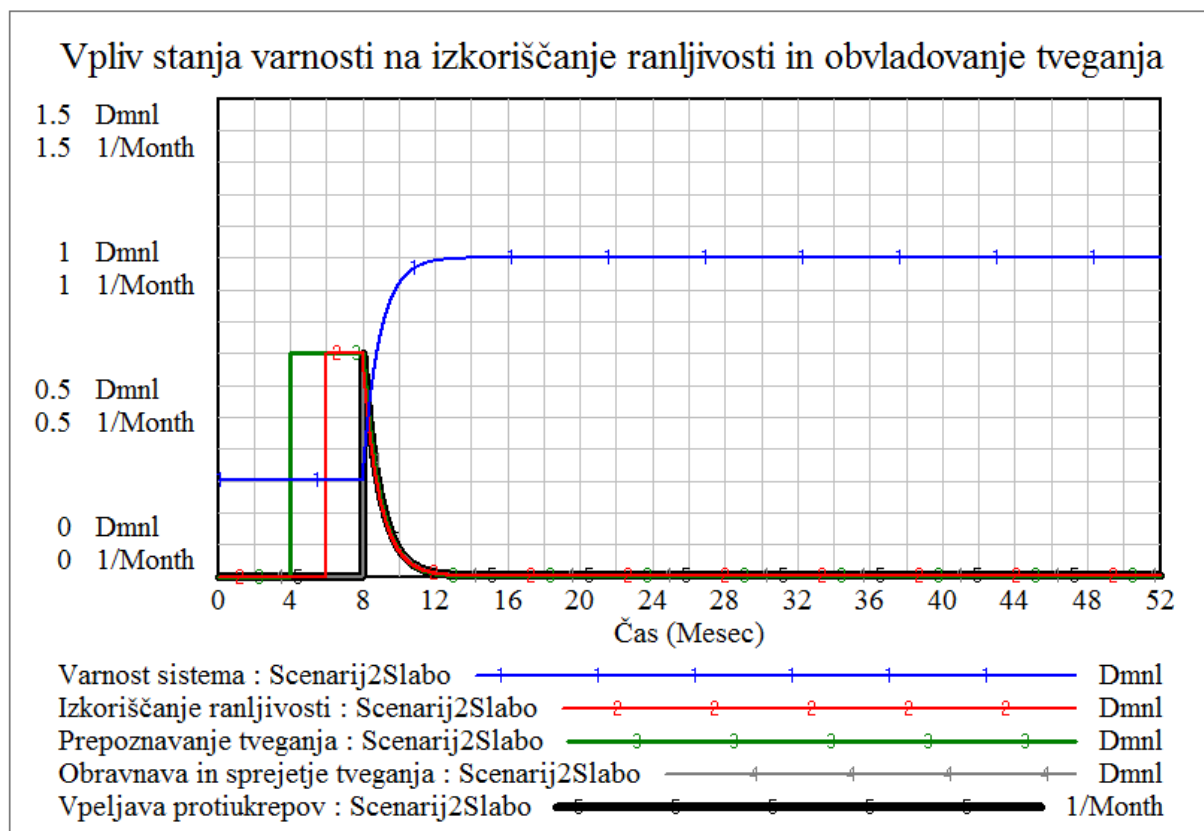
Srednja kakovost obvladovanja tveganja

Pri srednji kakovosti obvladovanja tveganja so spremenjeni privzeti začetni pogoji modela. Na spremembo začetnih pogojev vpliva uporaba orodja in metode pri obvladovanju tveganja in za zagotavljanje varnosti. V spodnji tabeli se nahajajo spremenjene vrednosti z dodanim komentarjem vzroka sprememb.

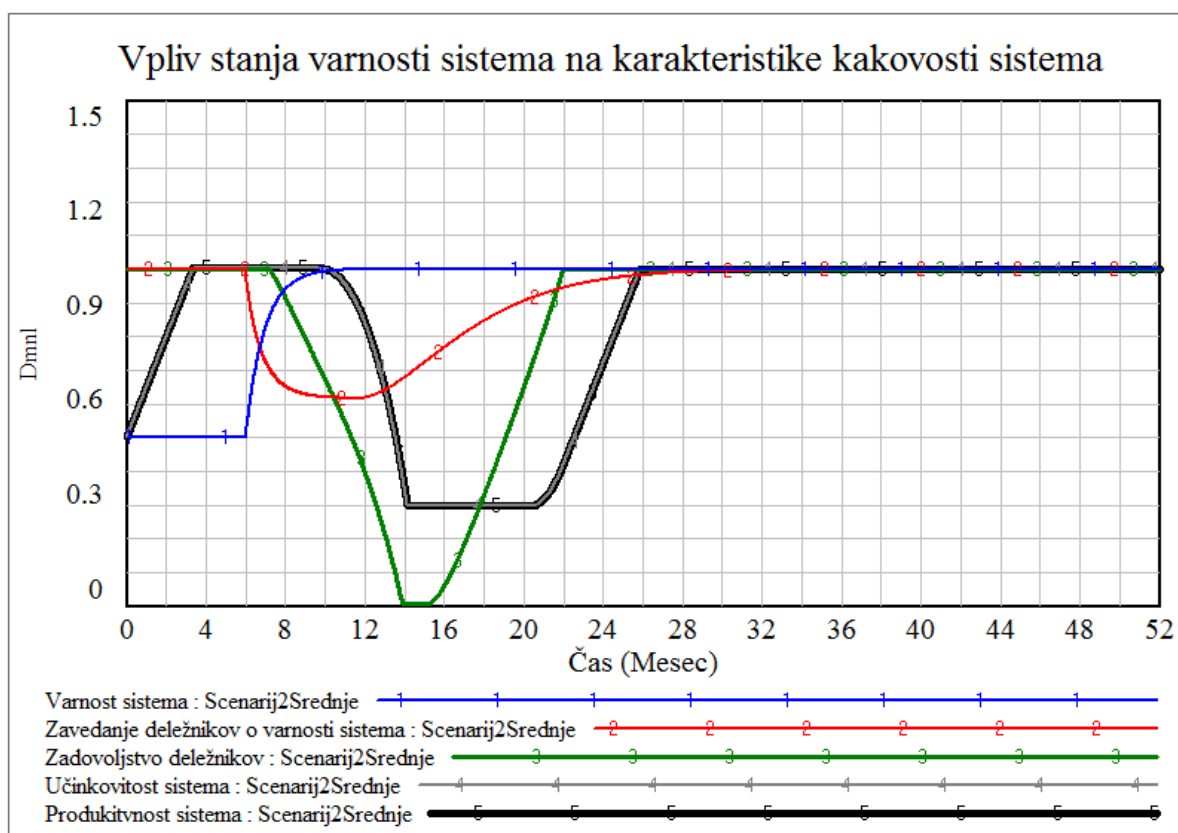
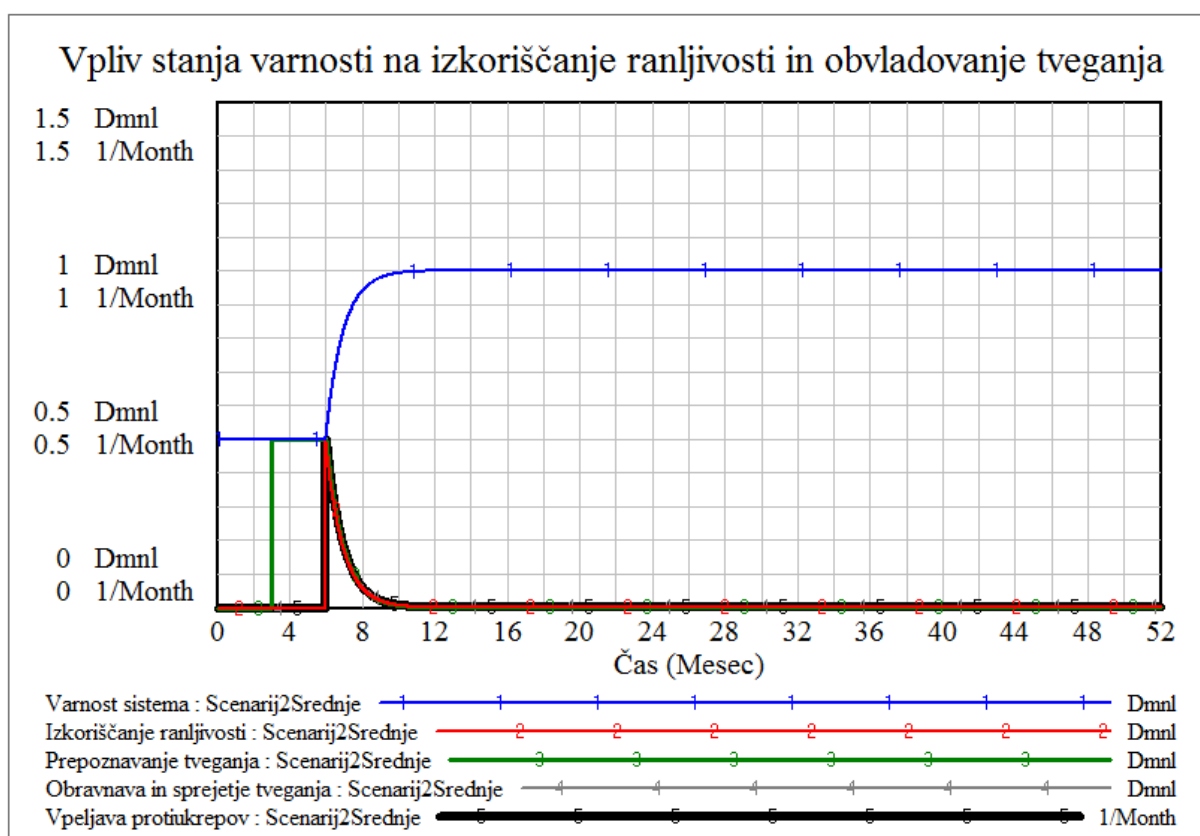
Spremenljivka	Vrednost	Razlog za spremembo
Čas izboljšanja zavedanja deležnikov o varnosti sistema	12 → 9 mesecev	Uporaba orodja in metode pripomore k bolj prepričljivemu zagotavljanju varnostnih funkcionalnosti. Posledično se zmanjša čas za izboljšanje zavedanja deležnikov o varnosti sistema, saj več deležnikov verjame v varnost sistema.
Čas prepoznavanja tveganja	5 → 3 mesecev	Uporaba strukturiranega pristopa zmanjša čas prepoznavanja tveganja.
Čas obravnave in sprejetja tveganja	5 → 3 mesecev	Uporaba strukturiranega pristopa zmanjša čas obravnave in sprejetja tveganja.
Začetna varnost sistema	30 % → 50 %	Uporaba orodja in metode pri zasnovi in razvoju sistema je strukturirani pristop za prepoznavanje ranljivosti. V primerjavi s pristopom "ad hoc" je večja verjetnost odkritja ranljivosti, kar pripomore k boljši začetni varnosti sistema.

Tabela 4.7: Začetni pogoji modela za scenarij "Nabava ranljivega sistema" — Srednja kakovost obvladovanja tveganja.

Diagrama na sliki 4.9 prikazujeta obnašanje modela. Začetna varnost sistema (zg. diag., črta 1) je boljša v primerjavi z začetno varnostjo slabe kakovosti obvladovanja tveganja. Dinamika izkoriščanja ranljivosti in vpeljava protiukrepov sovpadata (zg. diag., črta 2 in 5). Okno izkoriščanja (zg. diag, vrzel črte 1) je ožje v primerjavi s prejšnjim izidom, zato je škoda manjša (zg. diag., površina pod črto 2). Posledično je manjši padec zavedanja deležnikov o varnosti sistema (sp. diag., črta 2), kar ima pozitiven učinek na zadovoljstvo deležnikov (sp. diag., črta 3), učinkovitost sistema (sp. diag., črta 4) in produktivnost sistema (sp. diag., črta 5). Padec učinkovitosti in produktivnosti imata negativne ekonomske posledice za obdobje 17 mesecev. V primerjavi s prejšnjim primerom je izboljšanje 7 mesecev.



Slika 4.8: Scenarij “Nabava ranljivega sistema” — Slaba kakovost obvladovanja tveganja.



Slika 4.9: Scenarij "Nabava ranljivega sistema" – Srednja kakovost obvladovanja tveganja.

Dobra kakovost obvladovanja tveganja

Pri dobri kakovosti obvladovanja tveganja so spremenjeni privzeti začetni pogoji modela. Orodje in metoda sta izkoriščena na najboljši način.

Spremenljivka	Vrednost	Razlog za spremembo
Čas izboljšanja zavedanja deležnikov o varnosti sistema	12 → 6 mesecev	Uporaba orodja in metode pripomore k bolj prepričljivemu zagotavljanju varnostnih funkcionalnosti. Posledično se zmanjša čas za izboljšanje zavedanja deležnikov o varnosti sistema, saj več deležnikov verjame v varnost sistema.
Čas prepoznavanja tveganja	5 → 2 mesecev	Uporaba strukturiranega pristopa zmanjša čas zaznave tveganja.
Čas obravnave in sprejetja tveganja	5 → 2 mesecev	Uporaba strukturiranega pristopa zmanjša čas obravnave in sprejetja tveganja.
Začetna varnost sistema	30 % → 60 %	Uporaba orodja in metode pri zasnovi in razvoju sistema je strukturirani pristop za prepoznavanje ranljivosti. V primerjavi s pristopom "ad hoc" je večja verjetnost odkritja ranljivosti, kar pripomore k boljši začetni varnosti sistema.

Tabela 4.8: Začetni pogoji modela za scenarij "Nabava ranljivega sistema" – Dobra kakovost obvladovanja tveganja.

Diagrama na sliki 4.10 prikazujeta obnašanje modela. Začetna varnost sistema (zg. diag., črta 1) je najboljša v primerjavi z začetno varnostjo slabe in srednje kakovosti obvladovanja tveganja. Vse faze obvladovanja tveganja prehitevajo izkoriščanje incidentov (zg. diag., črta 2): (i) prepoznavanje tveganja (zg. diag., črta 3), (ii) obravnava in sprejetje tveganja (zg. diag., črta 4) ter (iii) vpeljava protiukrepov (zg. diag., črta 5). Okno izkoriščanja ranljivosti je najmanjše (zg. diag., vrzel črte 1). Škoda je minimalna (zg. diag., površina pod črto 2), saj je večina protiukrepov vpeljana pravočasno (zg. diag., črta 5). Majhna škoda povzroči zanemarljiv padec zavedanja deležnikov o varnosti sistema (sp. diag., črta 2) in nima učinka na zadovoljstvo deležnikov (sp. diag., črta 3), učinkovitost sistema (sp. diag., črta 4) in produktivnost sistema (sp. diag., črta 5). Zaradi dobrega obvladovanja tveganja ni negativnih ekonomskih posledic.

Primerjava izidov

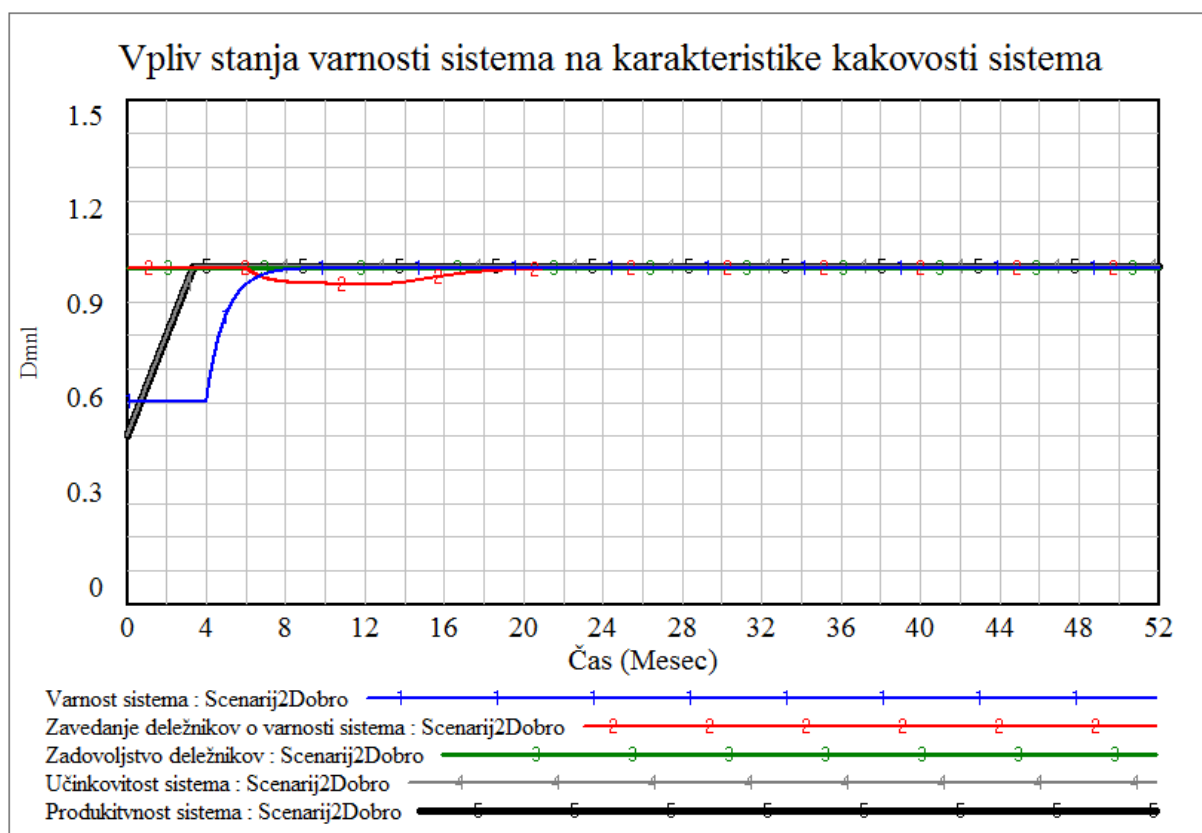
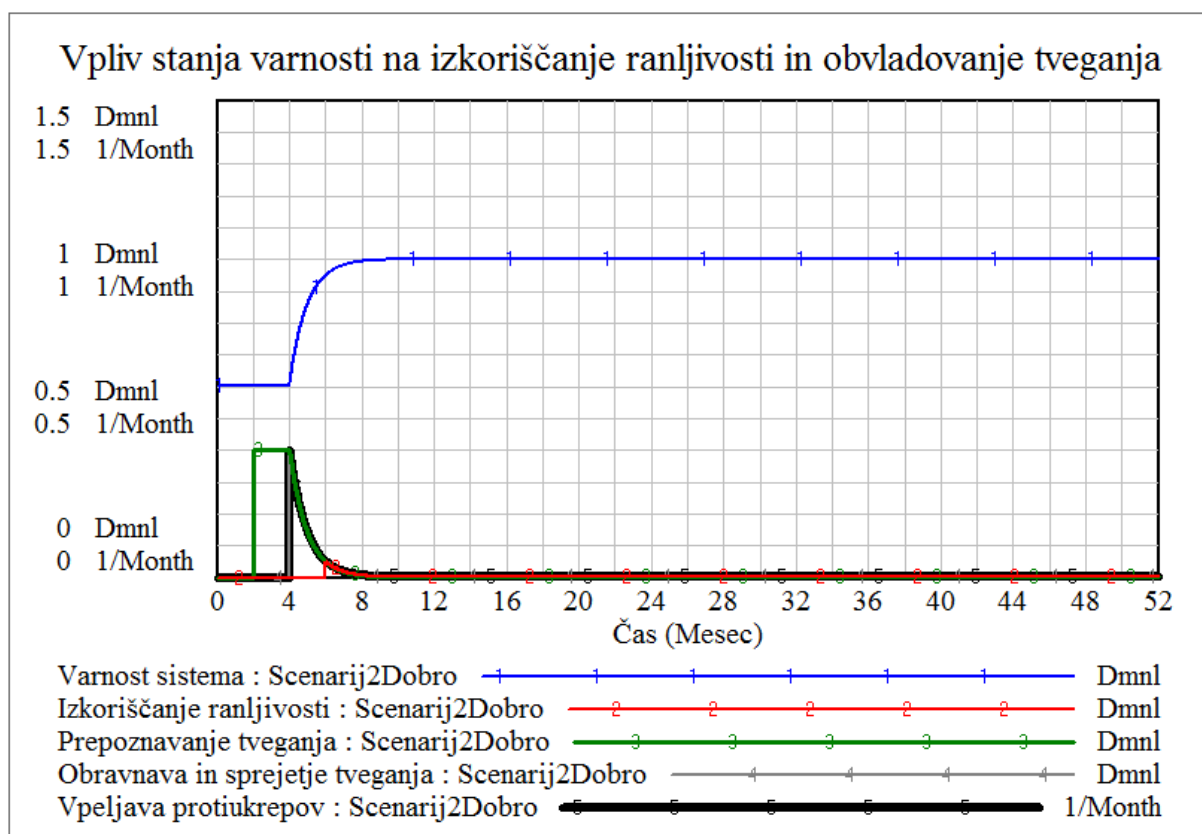
Na sliki 4.11 so zbrane primerjave vseh treh izidov scenarija "Nabava ranljivega sistema". Prožilec je v vseh izidih scenarija nabava ranljivega sistema. Stopnja ranljivosti se razlikuje in je odvisna od uporabe orodja in metode pri zasnovi, razvoju in nabavi sistema.

Vsi izidi scenarijev uporabljajo proaktivni pristop obvladovanja tveganja, kjer so tveganja prepoznana pred dogodkom incidenta. V najboljšem primeru so protiukrepi vpeljeni pred

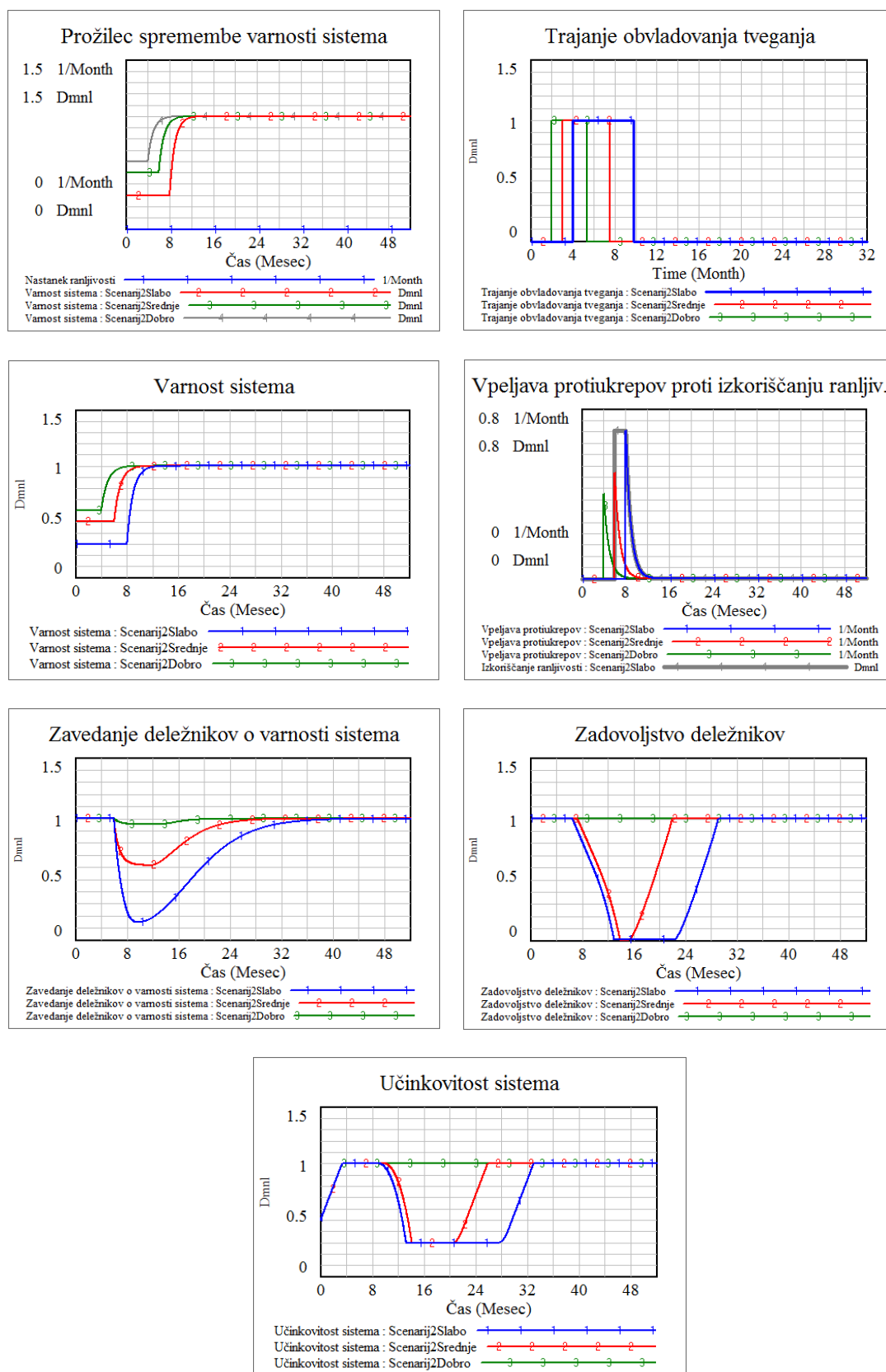
razkritjem ranljivosti hekerjev. Izrabljanje ranljivosti in s tem povezana škoda je odvisna od učinkovitosti obvladovanja tveganja, in sicer od:

- uporabe orodja in metode pri zasnovi, razvoju in nabavi sistema,
- uporabe orodja in metode v procesu obvladovanja tveganja.

V realnosti razvijalci navadno ne preizkušajo varnosti sistema, združbe ne poznajo varnostnih zahtev poslovnega sistema in imajo nezadostne zahteve o stopnji zanesljivosti ocene o varnostnih funkcionalnostih sistema zaradi nepoznavanja stopnje tveganja poslovnega sistema. Našteti dejavniki so v kombinaciji z reaktivnim pristopom obvladovanja tveganja, ki je neučinkovit, razlogi za velikansko škodo, ki jo utrpijo deležniki v primeru incidenta. Scenarij reaktivnega pristopa obvladovanja tveganja ni prikazan na diagramih slike 4.11. Škoda bi bila še večja od najslabšega prikazanega izida.



Slika 4.10: Scenarij “Nabava ranljivega sistema” – Dobra kakovost obvladovanja tveganja.



Slika 4.11: Primerjava obnašanja spremenljivk sistema treh izidov scenarija “Nabava ranljivega sistema”.

4.4.5 Nabava ranljivega sistema z nastankom nove ranljivosti pri aktivni uporabi

V tretjem scenariju nas bo zanimala dinamika kakovosti sistema z dvema prožilcema:

1. Prvi prožilec predstavlja ranljivost, ki je prisotna še pred postavitvijo sistema. Lahko je posledica nekovostne zasnove in razvoja sistema ali pa slabe presoje vodilnih pri nakupu sistema.
2. Drugi prožilec predstavlja ranljivost, ki nastane pri aktivni uporabi sistema.

V tretjem scenariju se začetno zavedanje deležnikov o varnosti sistema in dejanska začetna varnost razlikujeta. Deležniki in združba imajo preoptimistično prepričanje o varnosti sistema. Za ta scenarij so spremenjeni nekateri začetni pogoji modela (glej tabelo 4.3). Spremembe in opisi sprememb so navedeni v tabeli 4.9.

Spremenljivka	Vrednost	Opis
Začetna varnost sistema	30 %	Slaba začetna varnost.
Začetno zavedanje deležnikov o varnosti sistema	100 %	Slabo, saj se razlikuje od dejanske začetne varnosti sistema.

Tabela 4.9: Privzeti začetni pogoji modela za scenarij “Nabava ranljivega sistema z nastankom nove ranljivosti”.

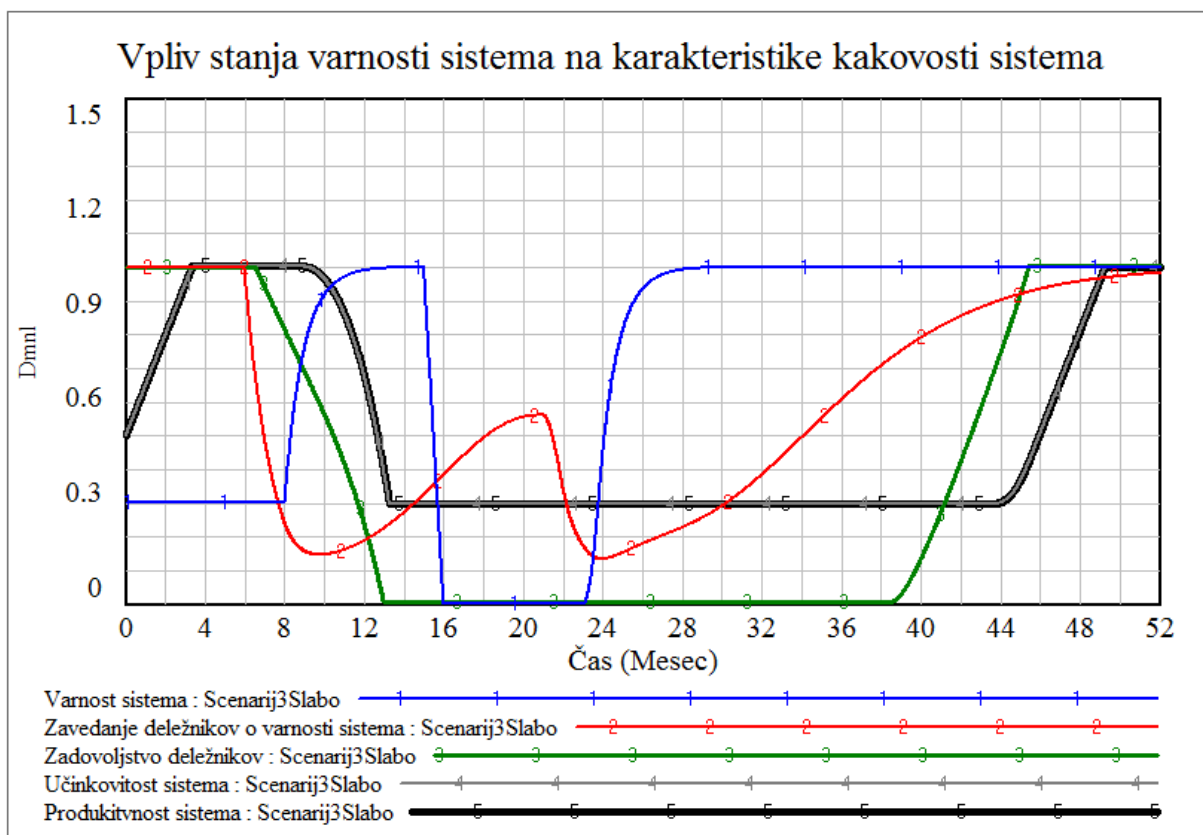
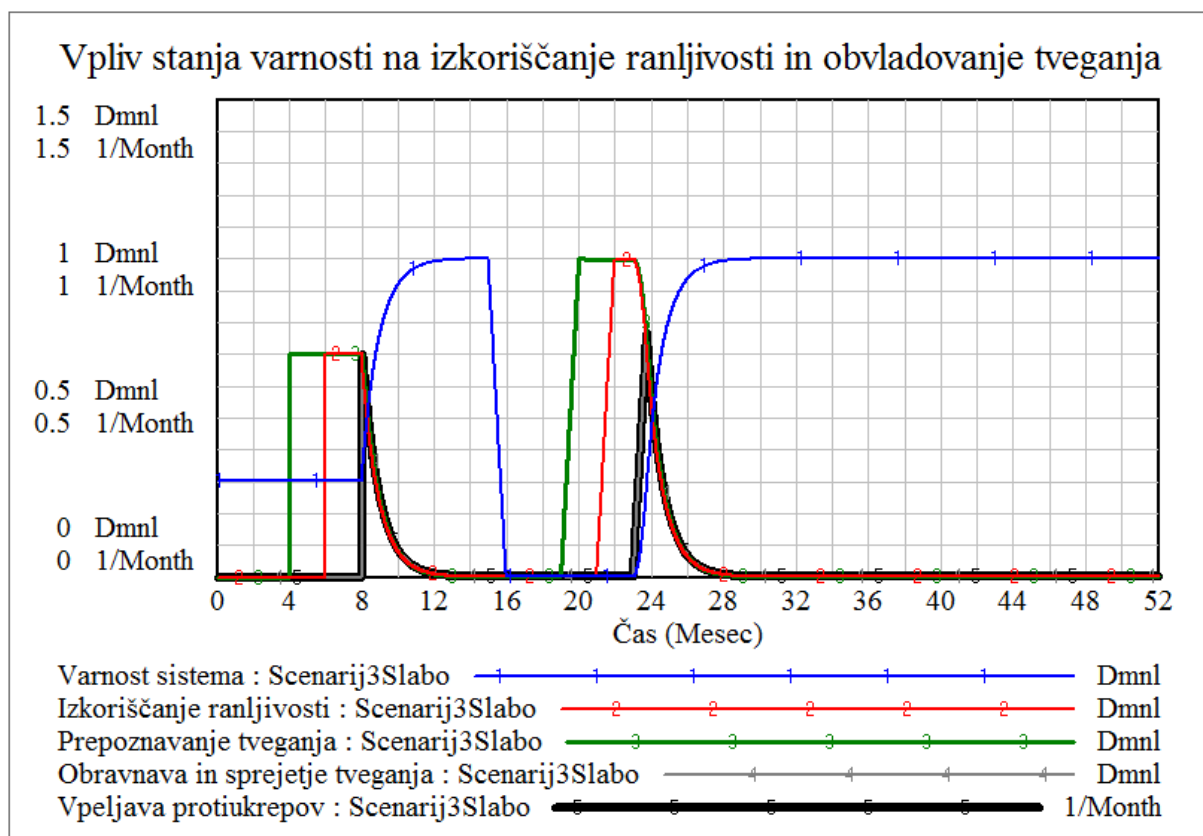
Slaba kakovost obvladovanja tveganja

Pri slabi kakovosti obvladovanja tveganja so uporabljeni privzeti začetni pogoji tabele 4.9. Diagrama na sliki 4.12 prikazujeta kompleksno obnašanje modela, ki je razdeljeno na dve obdobji. V obdobju od 0 do 15 mesecev poteka obvladovanje tveganja zaradi nabave ranljivega sistema, v obdobju od 15 do 52 mesecev pa poteka obvladovanje tveganja zaradi nove ranljivosti pri aktivni uporabi sistema. Zanima nas skupni učinek izkoriščanja ranljivosti na karakteristike kakovosti sistema.

Na zgornjem diagramu je prikazano stanje varnosti (zg. diag., črta 1), katerega začetna vrednost je slaba in znaša 30 %. Tako slaba vrednost začetne varnosti izhaja iz nekovostne zasnove in razvoja sistema ali pa je posledica slabe presoje vodilnih združb pri nakupu sistema. Hekerji začnejo izkoriščati ranljivost po preteku časa razkritja ranljivosti (zg. diag., črta 4). Posledica izkoriščanja je škoda, ki jo občutijo deležniki. Sledi prilagoditev zavedanja deležnikov o varnosti sistema (sp. diag., črta 2). Odziv prve faze obvladovanja tveganja, tj. prepoznavanje tveganja, prehiteva izkoriščanje ranljivosti (zg. diag., črta 2 in 3). Kljub temu nastane škoda zaradi izkoriščanja ranljivosti, saj so protiukrepi vpeljeni prepozno zaradi slabe učinkovitosti procesa obvladovanja tveganja (zg. diag., črta 4 in 5). Okno izkoriščanja ranljivosti je široko (zg. diag., vrzel črte 1), zato je povzročena škoda velika (zg. diag., površina pod črto 2). Izkoriščanje se ustavi, ko so vpeljeni vsi protiukrepi (zg. diag., črta 5) v 10. mesecu.

V 15. mesecu se pojavi nova ranljivost. Stanje varnosti se takoj zmanjša na 0 %. Hekerji začnejo izkoriščati ranljivosti po preteku časa razkritja ranljivosti (zg. diag., črta 4). Posledica izkoriščanja je prilagoditev zavedanja deležnikov o varnosti sistema, ki kasni glede na čas nastanka ranljivosti (sp. diag., črta 1 in črta 2). Odziv prve faze obvladovanja tveganja, tj. prepoznavanje tveganja, prehiteva izkoriščanje ranljivosti (zg. diag., črta 2 in 3), vendar so protiukrepi vpeljeni prepozno zaradi slabe učinkovitosti obvladovanja tveganja (zg. diag., črta 4 in 5). Okno izkoriščanja ranljivosti je široko (zg. diag., vrzel črte 1), zato je povzročena škoda velika (zg. diag., površina pod črto 2). Izkoriščanje ranljivosti se ustavi, ko so vpeljeni vsi protiukrepi (zg. diag., črta 5) v 28. mesecu.

Deležniki daljši čas niso prepričani v pravilnost in učinkovitost protiukrepov zaradi pristopa "ad hoc" pristopa v obeh obdobjih obvladovanja tveganja. Posledica tega je, da se varnost sistema in zavedanje deležnikov o varnosti sistema daljši čas razlikuje (sp. diag., črta 1 in 2). Medtem ko se zavedanje deležnikov o varnosti sistema popravlja, v 21. mesecu sledi nov padec zavedanja zaradi izkoriščanja nove ranljivosti. To povzroči dolgotrajno nezadovoljstvo (sp. diag., črta 3), ki ima kaskadni vpliv tudi na produktivnost in učinkovitost sistema (sp. diag., črta 4 in 5). Negativne ekonomske posledice trajajo za trajanje obeh obdobj 40 mesecev.



Slika 4.12: Scenarij “Nabava ranljivega sistema z nastankom nove ranljivosti” – Slaba kakovost obvladovanja tveganja.

Srednja kakovost obvladovanja tveganja

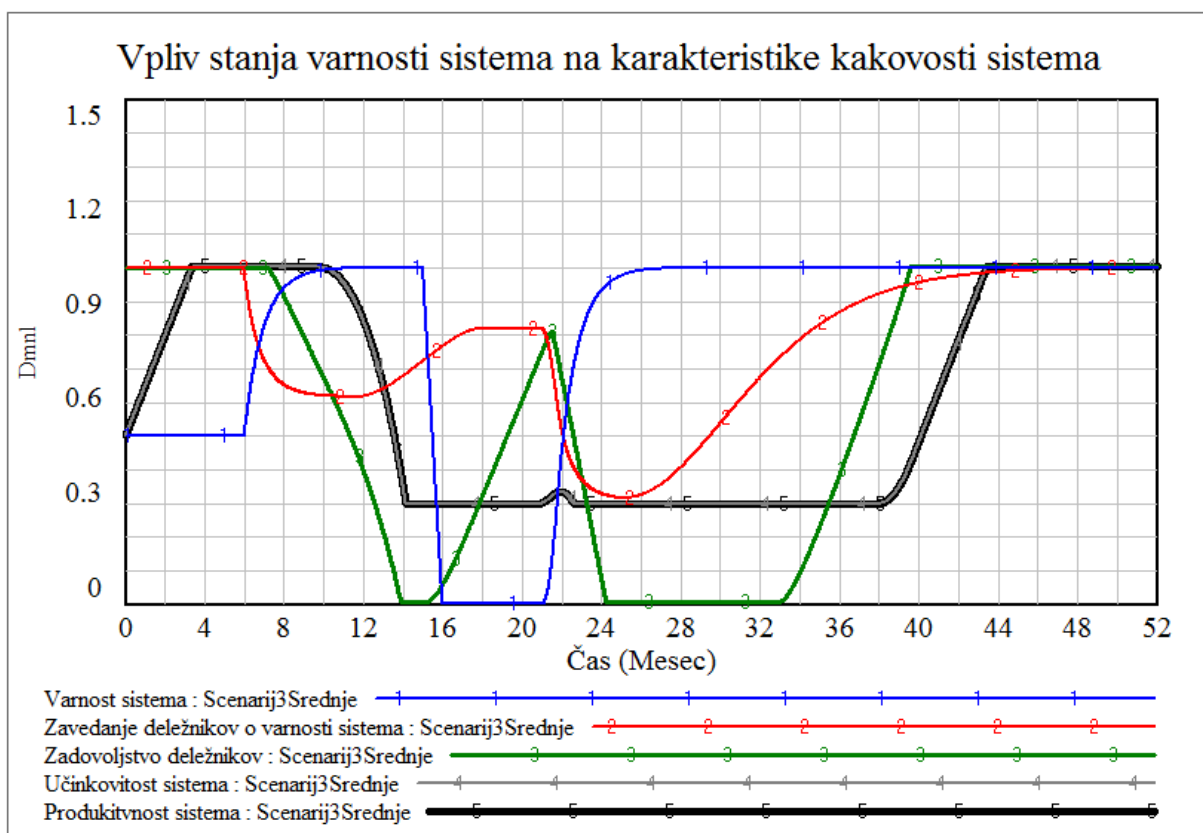
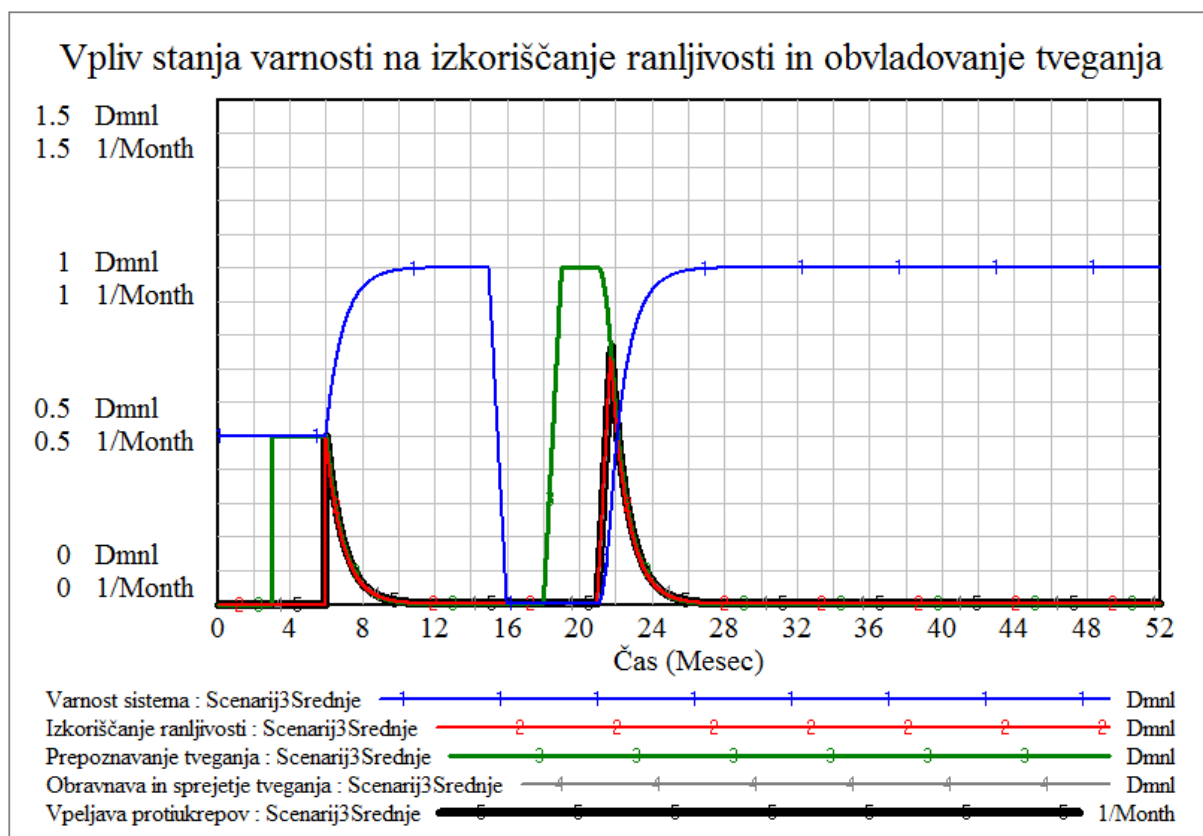
Pri srednji kakovosti obvladovanja tveganja so spremenjeni privzeti začetni pogoji modela. Na spremembo začetnih pogojev vpliva uporaba orodja in metode pri obvladovanju tveganja in za zagotavljanje varnosti. V spodnji tabeli se nahajajo spremenjene vrednosti z dodanim komentarjem vzroka sprememb.

Spremenljivka	Vrednost	Razlog za spremembo
Čas izboljšanja zavedanja deležnikov o varnosti sistema	12 → 9 mesecev	Uporaba orodja in metode pripomore k bolj prepričljivemu zagotavljanju varnostnih funkcionalnosti. Posledično se zmanjša čas za izboljšanje zavedanja deležnikov o varnosti sistema, saj več deležnikov verjame v varnost sistema.
Čas prepoznavanja tveganja	5 → 3 mesecev	Uporaba strukturiranega pristopa zmanjša čas prepoznavanja tveganja.
Čas obravnave in sprejetja tveganja	5 → 3 mesecev	Uporaba strukturiranega pristopa zmanjša čas obravnave in sprejetja tveganja.
Začetna varnost sistema	30 % → 50 %	Uporaba orodja in metode pri zasnovi in razvoju sistema je strukturirani pristop za preizkušanje varnostnih zahtev. V primerjavi s pristopom "ad hoc" so varnostne zahteve preizkušene bolj obširno, kar zviša verjetnost odkritja ranljivosti pred postavitvijo sistema.

Tabela 4.10: Začetni pogoji modela za scenarij "Nabava ranljivega sistema z nastankom nove ranljivosti" — Srednja kakovost obvladovanja tveganja.

Diagrama na sliki 4.13 prikazujeta kompleksno obnašanje modela, ki je razdeljeno na dve obdobji. V obdobju od 0 do 15 mesecev poteka obvladovanje tveganja zaradi nabave ranljivega sistema, v obdobju od 15 do 52 mesecev pa poteka obvladovanje tveganja zaradi nove ranljivosti pri aktivni uporabi sistema. Zanima nas skupni učinek izkoriščanja ranljivosti na karakteristike kakovosti sistema.

Začetna varnost sistema (zg. diag., črta 1) je boljša v primerjavi z začetno varnostjo slabe kakovosti obvladovanja tveganja. Dinamika izkoriščanja ranljivosti in vpeljava protiukrepov sovpadata v obeh obdobjih (zg. diag, črta 2 in 5). Okno izkoriščanja (zg. diag., vrzel črte 1) je ožje v primerjavi s prejšnjim izidom, zato je škoda v obeh obdobjih manjša (zg. diag., površina pod črto 2). Posledično je manjši padec zavedanja deležnikov o varnosti sistema (sp. diag., črta 2), kar ima pozitiven učinek na zadovoljstvo deležnikov (sp. diag., črta 3), učinkovitost sistema (sp. diag., črta 4) in produktivnost sistema (sp. diag., črta 5). Padec učinkovitosti in produktivnosti ima negativne ekonomske posledice za obdobje 33 mesecev. V primerjavi s prejšnjim primerom je izboljšanje za 7 mesecev.



Slika 4.13: Scenarij “Nabava ranljivega sistema z nastankom nove ranljivosti” — Srednja kakovost obvladovanja tveganja.

Dobra kakovost obvladovanja tveganja

Pri dobri kakovosti obvladovanja tveganja so spremenjeni začetni pogoji modela. Metoda in orodje sta izkoriščena na naboljši način.

Spremenljivka	Vrednost	Razlog za spremembo
Čas izboljšanja zavedanja deležnikov o varnosti sistema	12 → 6 mesecev	Uporaba orodja in metode pripomore k bolj prepričljivemu zagotavljanju varnostnih funkcionalnosti. Posledično se zmanjša čas za izboljšanje zavedanja deležnikov o varnosti sistema, saj več deležnikov verjame v varnost sistema.
Čas prepoznavanja tveganja	5 → 2 mesecev	Uporaba strukturiranega pristopa zmanjša čas prepoznavanja tveganja.
Čas obravnave in sprejetja tveganja	5 → 2 mesecev	Uporaba strukturiranega pristopa zmanjša čas obravnave in sprejetja tveganja.
Začetna varnost sistema	30 % → 60 %	Uporaba orodja in metode pri zasnovi in razvoju sistema je strukturirani pristop za preizkušanje varnostnih zahtev. V primerjavi s pristopom "ad hoc" so varnostne zahteve preizkušene bolj obširno, kar zviša verjetnost odkritja ranljivosti pred postavitvijo sistema.

Tabela 4.11: Začetni pogoji modela za scenarij "Nabava ranljivega sistema z nastankom nove ranljivosti" — Dobra kakovost obvladovanja tveganja.

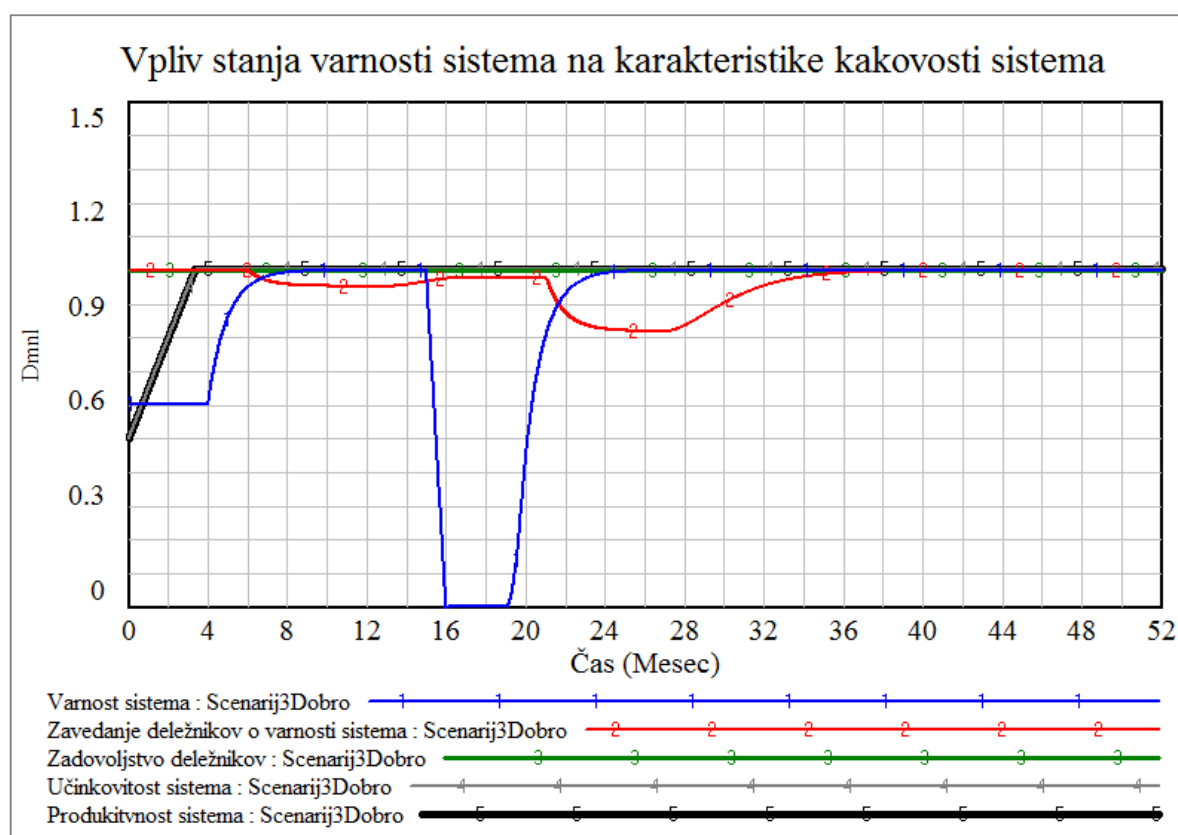
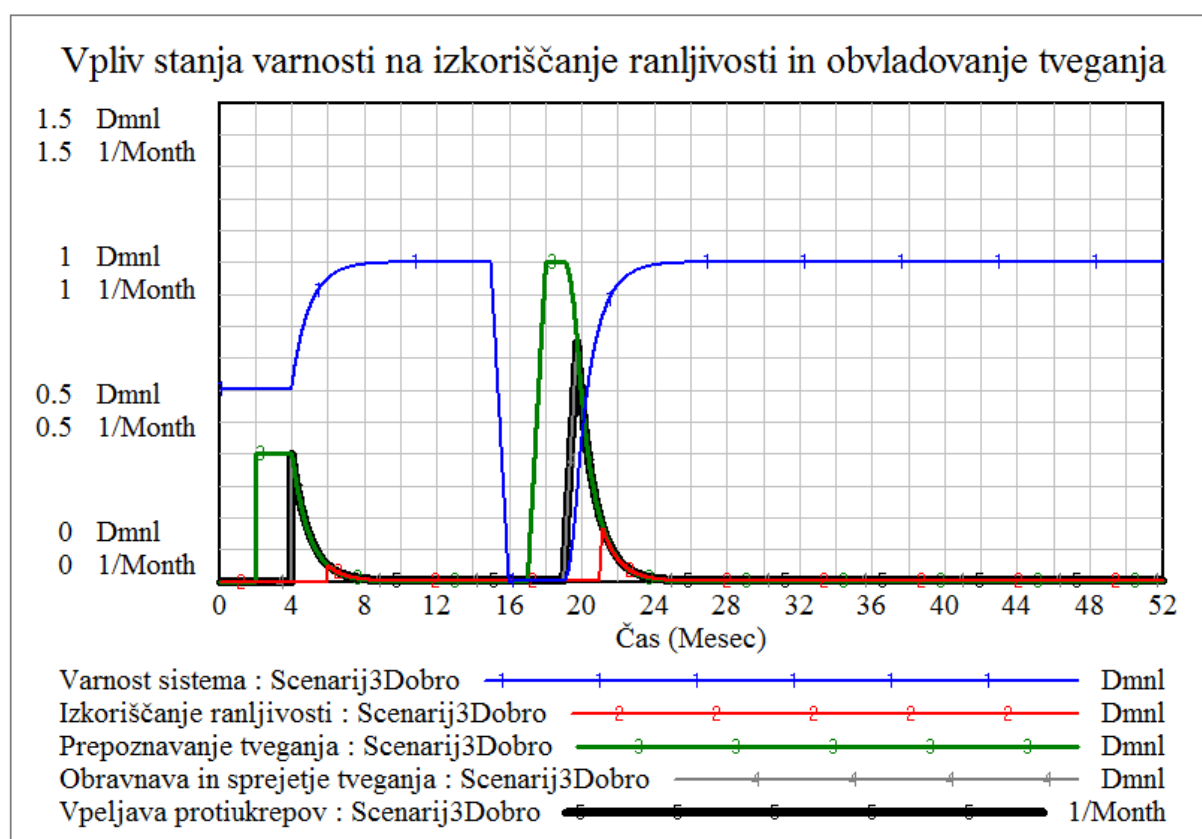
Diagrama na sliki 4.14 prikazujeta kompleksno obnašanje modela, ki je razdeljeno na dve obdobji. V obdobju od 0 do 15 mesecev poteka obvladovanje tveganja zaradi nabave ranljivega sistema, v obdobju od 15 do 52 mesecev pa poteka obvladovanje tveganja zaradi nove ranljivosti pri aktivni uporabi sistema. Zanima nas skupni učinek izkoriščanja ranljivosti na karakteristike kakovosti sistema.

Začetna varnost sistema (zg. diag, črta 1) je najboljša v primerjavi z začetno varnostjo slabe in srednje kakovosti obvladovanja tveganja. Vse faze obvladovanja tveganja preprečujejo izkoriščanje incidentov v obeh obdobjih (zg. diag., črta 2): prepoznavanje tveganja (zg. diag., črta 3), obravnava in sprejetje tveganja (zg. diag., črta 4) ter vpeljava protiukrepov (zg. diag., črta 5). Okno izkoriščanja ranljivosti je najmanjše (zg. diag, vrzel črte 1). Škoda je minimalna (zg. diag., površina pod črto 2), saj je večina protiukrepov vpeljana pravočasno (zg. diag., črta 5). Majhna škoda povzroči zanemarljiv padec zavedanja deležnikov o varnosti sistema v obeh obdobjih (sp. diag., črta 2). Zaradi tega sistem deluje pri polni učinkovitosti (sp. diag., črta 4) in produktivnosti sistema (sp. diag., črta 5). Zaradi dobrega obvladovanja tveganja ni negativnih ekonomskih posledic.

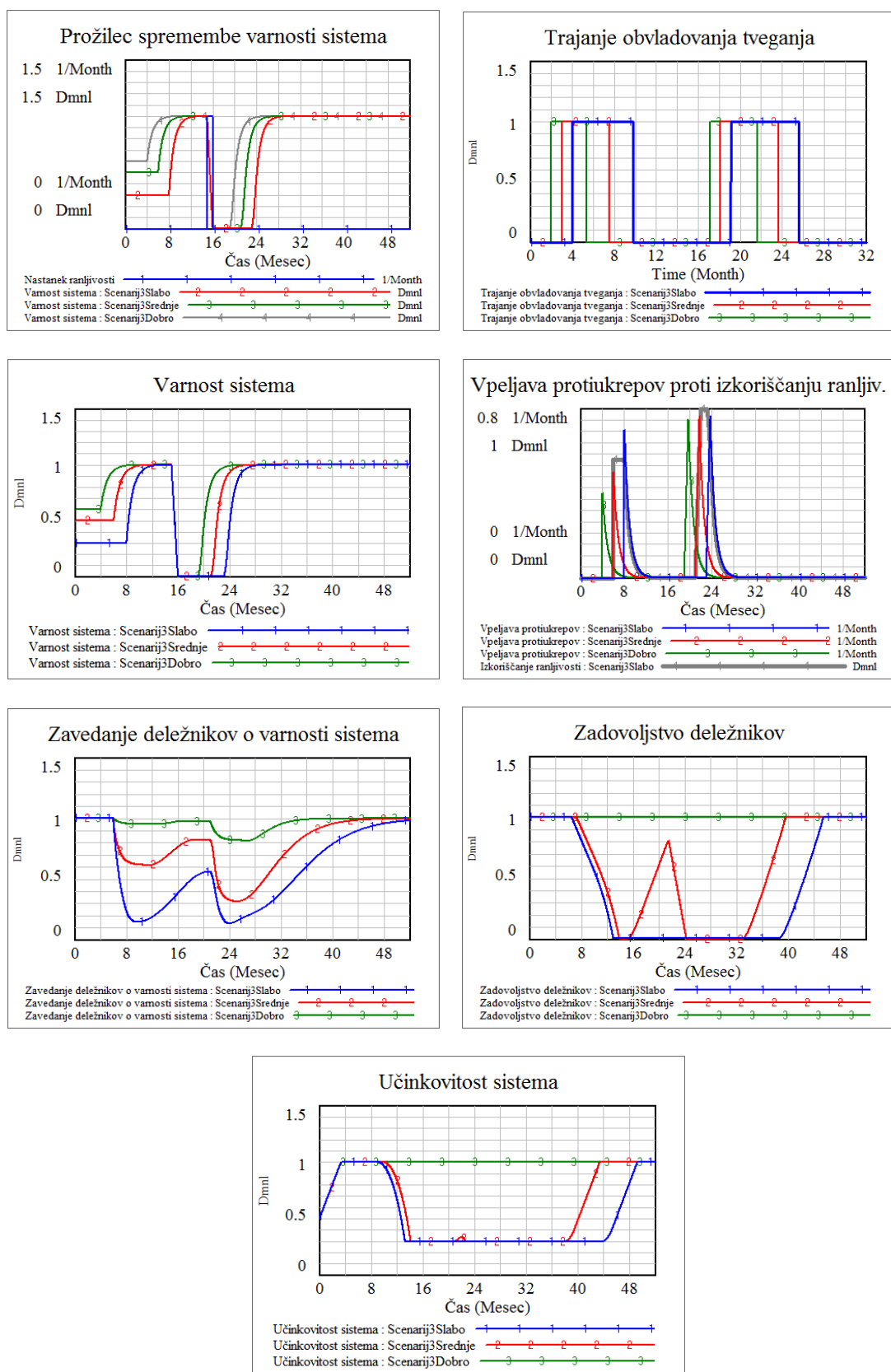
Primerjava izidov

Na sliki 4.15 so zbrane primerjave vseh treh izidov scenarija “Nabava ranljivega sistema z nastankom nove ranljivosti”. Prožilec je v vseh izidih scenarija nabava ranljivega sistema in nastanek nove ranljivosti pri aktivni uporabi sistema v času 15 meseca.

Vsi izidi scenarija uporabljajo proaktivni pristop obvladovanja tveganja, ker so tveganja prepoznana pred dogodkom incidenta. V najboljšem primeru so protiukrepi v večini vpeljani pred razkritjem ranljivosti hekerjev in škoda izkoriščanja ranljivosti je minimalna. V realnosti je pristop reaktiven. Ta izid ni prikazan na diagramih slike 4.15. Iz istih razlogov, kot so bili navedeni v prejšnjih dveh scenarijih, bi bila pri reaktivnem pristopu obvladovanja tveganja škoda večja od najslabše prikazanega izida.



Slika 4.14: Scenarij “Nabava ranljivega sistema z nastankom nove ranljivosti” — Dobra kakovost obvladovanja tveganja.



Slika 4.15: Primerjava obnašanja spremenljivk sistema treh izidov scenarija “Nabava ranljivega sistema z nastankom nove ranljivosti”.

OBRAVNAVALI smo področje varnosti za okolja različnih omrežnih sistemov radiofrekvenčne identifikacije. Pretekli dogodki incidentov pričajo o slabi varnosti sistemov, ki je nezadostno. Določeni delež vzroka za slabo stanje varnosti nosi nepoznavanje groženj. Drugi delež krivde pa nosi slabo prepoznavanje ranljivosti. Lastniki sistemov opravičujejo slabo stanje varnosti s pomanjkanjem pravih orodij za preizkušanje varnosti, uporabniki sistemov pa imajo napačen vtis, da je za varnost poskrbljeno v enaki meri kot za funkcionalnost. Uporabniki se zaradi tega šele po incidentu začnejo zavedati resnične varnosti sistema. To ima negativen vpliv na zadovoljstvo, posledično pa tudi na učinkovitost in produktivnost sistema. Dodatni problem predstavlja dejstvo, da se značke v veliki večini primerov ne da nadgraditi, ampak jo je potrebno fizično zamenjati.

Ustvarili smo nizkocenovno orodje, s katerim se izvedejo napadi na logiko značke in njeno komunikacijo s čitalcem ter prepoznajo njene ranljivosti. Ranljivosti značke vplivajo na varnost in kakovost celotnega sistema. Uporaba orodja pripomore k izboljšanju učinkovitosti in produktivnosti prepoznavanja ranljivosti značk. Prepoznavanje ranljivosti poteka na podlagi pravilno prepoznanih groženj sistemu, ki so specifične za vsak primer uporabe omrežnega sistema RFID posebej.

Definirana je bila metoda zagotavljanja varnosti, ki je osnovana na uporabi orodja v procesu obvladovanja tveganja pri varovanju informacij ISO/IEC 27005:2008. Uporaba metode v okoljih omrežnih sistemov RFID izboljša kakovost storitev sistema na naslednji način. Potrjene so ranljivosti značk in prepoznana tveganja so obravnavana. Če ranljivost ni prepoznana, uporaba metode pri prepoznavanju ranljivosti zagotovi določeno stopnjo varnosti sistema. Uporaba metode v okviru svojih zmožnosti omogoča spremembo pristopa obvladovanja tveganja z reaktivnega, ki sledi posledicam in predstavlja trenutno stanje, na proaktivno, ki prehiteva dogodke incidentov in jih preprečuje.

Zgornje trditve smo potrdili na naslednji način. Preučili smo mogoče grožnje okolij omrežnih sistemov RFID in izvedbe napadov na značke. Napade smo razčlenili na elemente. Orodje je učinkovito in sposobno izvesti vse elemente napadov. To smo pokazali s praktično demonstracijo. Produktivnost orodja smo analizirali s pomočjo tipologije lastnosti

porazdeljenih sistemov. Vpliv uporabe metode smo proučili s simulacijo. Izsledki simulacije so pokazali, da uporaba orodja pri obvladovanju tveganja pripomore k izboljšanju dinamike kakovosti sistema, in sicer na naslednji način.

- Čas prepoznavanja tveganja se zmanjša.
- Čas obravnave tveganja se zmanjša, potrdi se pravilnost in učinkovitost protiukrepov značke.
- Metoda zagotavljanja varnosti okolij omrežnih sistemov RFID je strukturirani pristop obvladovanja tveganja, ki zmanjša potrebni čas, da zainteresirane stranke okrepijo zaupanje v varnost sistema.
- Uporaba metode zagotavljanja varnosti okolij omrežnih sistemov RFID pri razvoju značke, pripomore k zgodnji prepoznavi in odpravi ranljivosti.

Orodje in metoda imata potencial za uporabo v industriji.

- Deležniki imajo vedno večje zahteve varnosti in zasebnosti pri uporabi tehnologije RFID. Orodje in metoda zagotavljanja varnosti je strukturirani pristop za preizkušanje ranljivosti in v primerjavi s pristopom “ad hoc” predstavlja boljše zagotovilo deležnikom, da so njihove varnostne zahteve zagotovljene.
- Združbe imajo potrebo po učinkovitem in produktivnem procesu obvladovanja tveganja. Orodje in metoda zagotavljanja varnosti je specializacija procesa obvladovanja tveganja ISO/IEC 27005:2008 za okolja omrežnih sistemov RFID s preizkušanjem značk RFID.
- Zgornja omejitev števila vrat značk se zvišuje. To omogoča uporabo boljših protiukrepov in uporabo boljših lahko-kriptografskih rešitev po meri. Te protiukrepe je potrebno preizkusiti, ali so primerni in učinkoviti ter ali vnesejo novo ranljivost v omrežni sistem RFID. Tudi omrežni sistemi RFID se hitro razvijajo in kontekst uporabe sistemov se širi. Z uporabo orodja in metode zagotavljanja varnosti je mogoče slediti opisani dinamiki sprememb na cenovno ugoden način.

5.1 Prispevki

Izvirni prispevki so sledeči:

1. Pregled groženj omrežnih sistemov RFID in možnih napadov na sistem z izkoriščanjem ranljivosti značk RFID.
2. Definicija orodja za preizkušanje varnosti sistema RFID s preizkušanjem značk RFID.

3. Definicija metode zagotavljanja varnosti, ki je specializacija procesa obvladovanja tveganja ISO/IEC 27005:2008 za okolja omrežnih sistemov RFID s preizkušanjem značk RFID. Uporaba orodja in metode izboljša učinkovitost prepoznavanja tveganja in pripomore k bolj prepričljivemu zagotavljanju varnostnih funkcionalnosti.

5.2 Nadaljnje delo

Predstavljeno delo ima številne možnosti za nadaljnje izboljšave, in sicer:

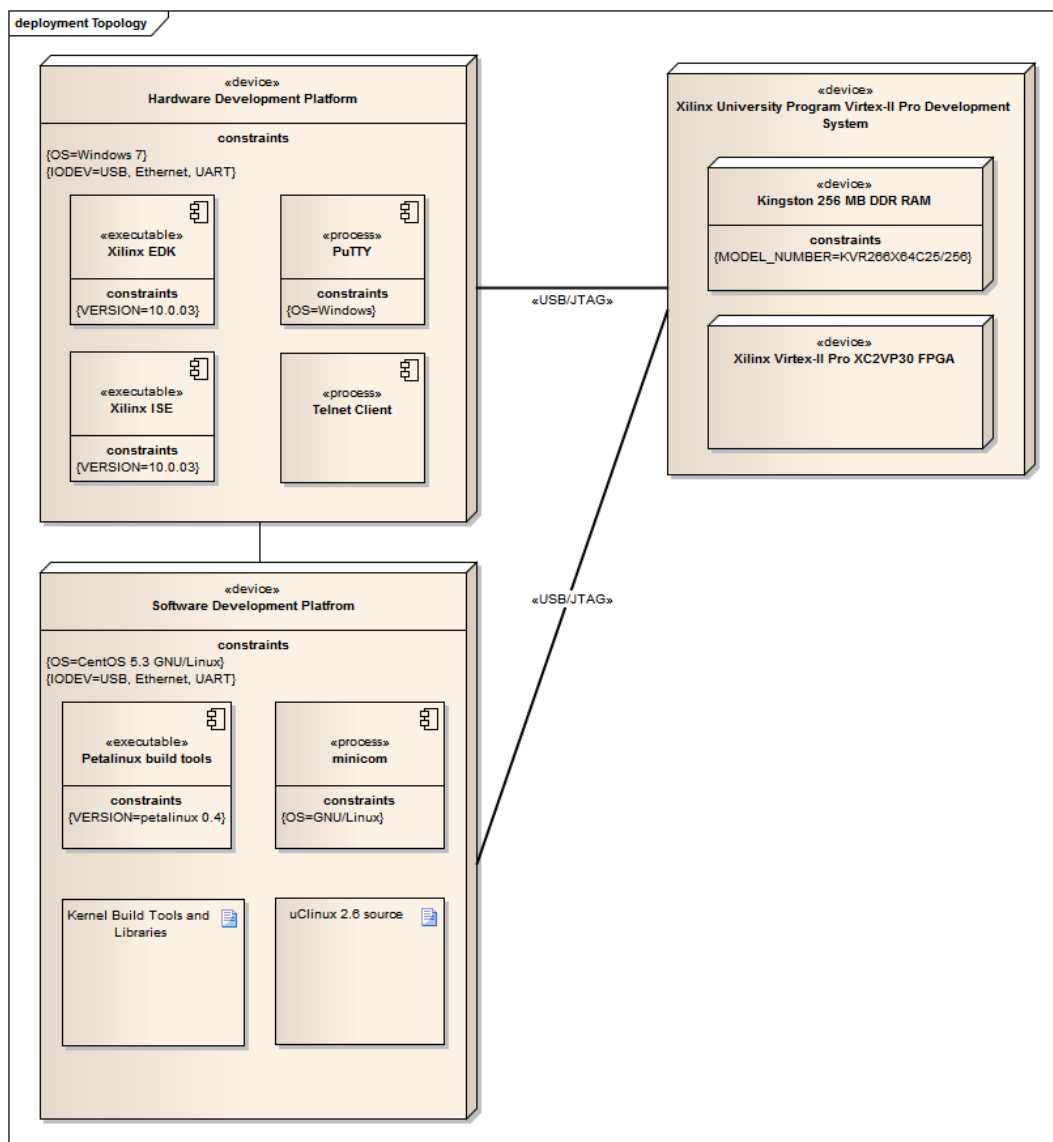
- Podpora nekaterih drugih karakteristik transparentnosti za boljšo produktivnost preizkušanja varnosti:
 1. Za večuporabniško interaktivno preizkušanje varnosti mora orodje podpirati transparentnost replikacije značke in transparentnost sočasne uporabe emulacijskega okolja.
 2. Podpora transparentnosti paralelnih aktivnosti bi omogočala avtomatizirano paralelno preizkušanje varnosti.
- Podpora za preizkušanja asinhronih digitalnih vezij bi razširila omejitev abstraktnega modela značke s sinhronega na asinhronega. Stanje vezja značke bi se pri tem moralo beležiti ob vsaki (asinhroni) spremembi.
- Trenutna realizacija značke je z emulacijo vezja FPGA. To podporno infrastrukturo bi nadomestili s simulacijsko opremo, ki upošteva fizikalne omejitve integriranih vezij. Odprtokodni simulatorji tega še ne omogočajo, lastniški pa so zaprti. Ta nadomestek bi omogočal replikacijo značk v oblaku in masivno paralelizacijo aktivnosti preizkušanja.
- Podpora za pametno avtomatizacijo testiranja.



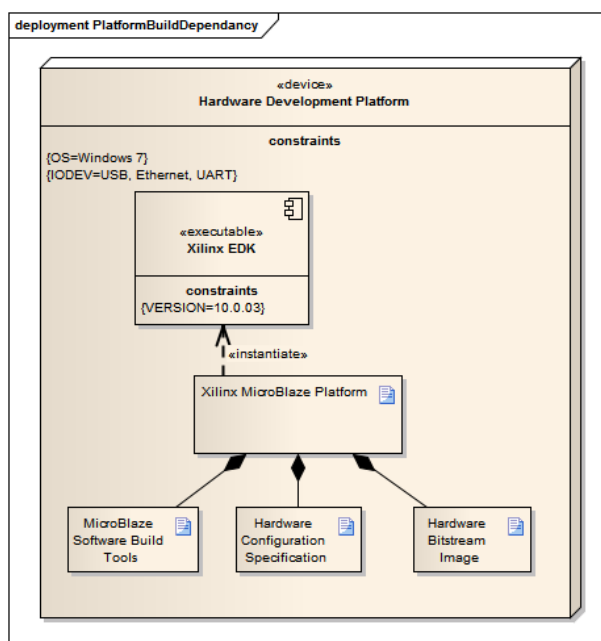
Zahteve za načrtovanje in razvoj orodja

Spodnji diagrami postavitve UML predstavljajo zahteve za načrtovanje in razvoj orodja in so bralcu v oporo in za orientacijo pri izvajanju tega zapletenega postopka. Vsebina diagramov je vzročno povezana. Posamezni elementi diagramov bodo podrobno obravnavani v naslednjih poglavjih.

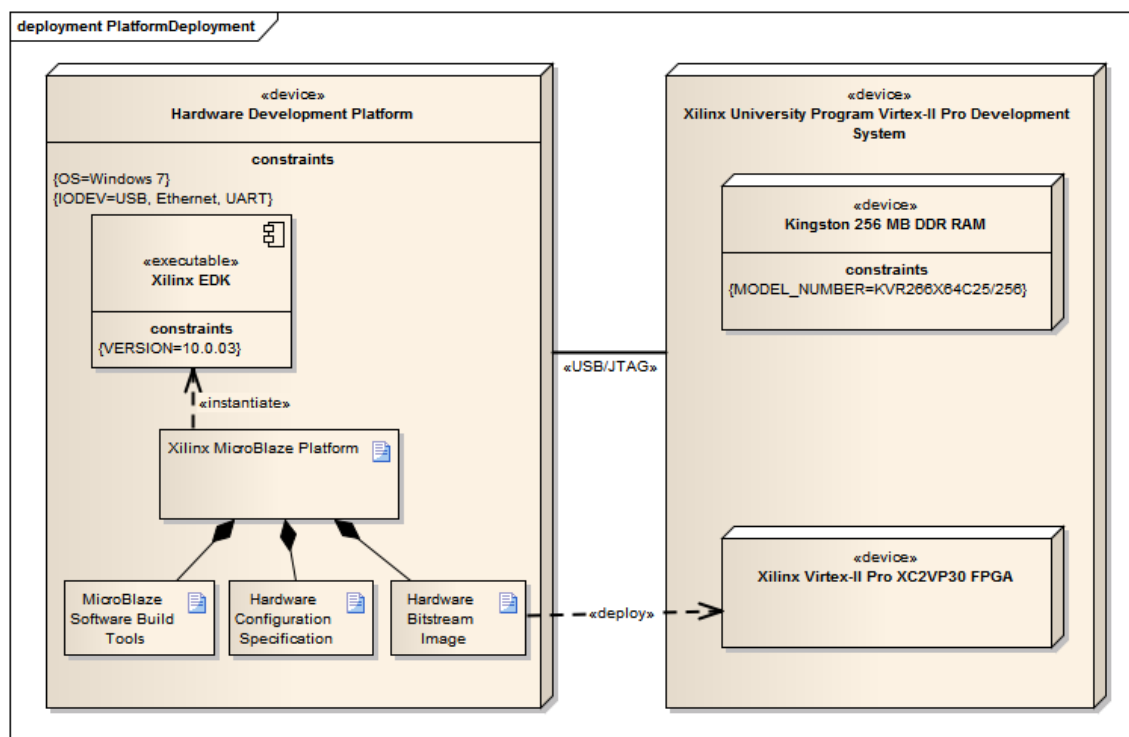
Seznam uporabljenih elementov in začetne topologije je prikazan na diagramu A.1, podrobno pa opisan v dodatku F.



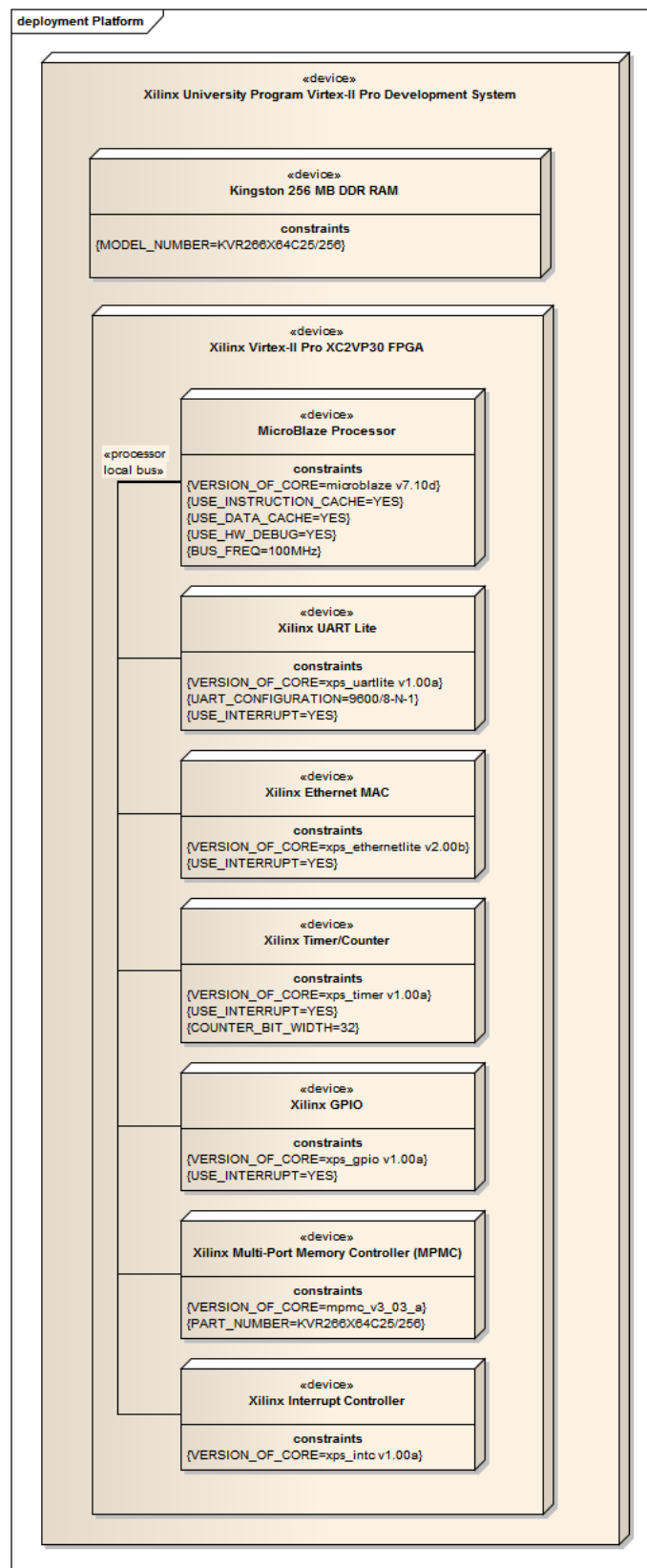
Slika A.1: Diagram postavitve uporabljene strojne in programske opreme



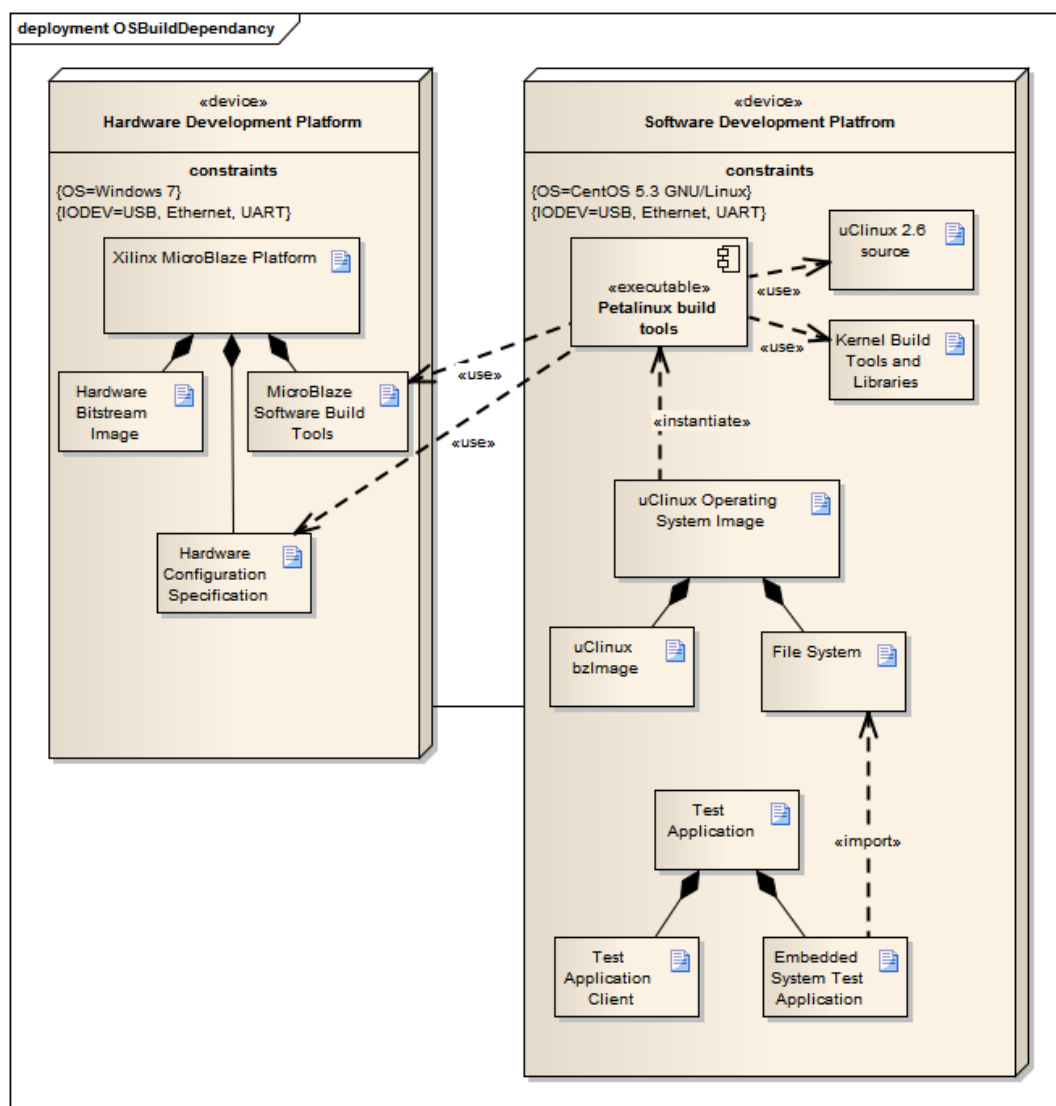
Slika A.2: Diagram postavitve UML: Definicija in gradnja strojne opreme ter osnovnih orodij za prevajanje in gradnjo programske opreme za ciljni sistem v razvojnem okolju Xilinx EDK



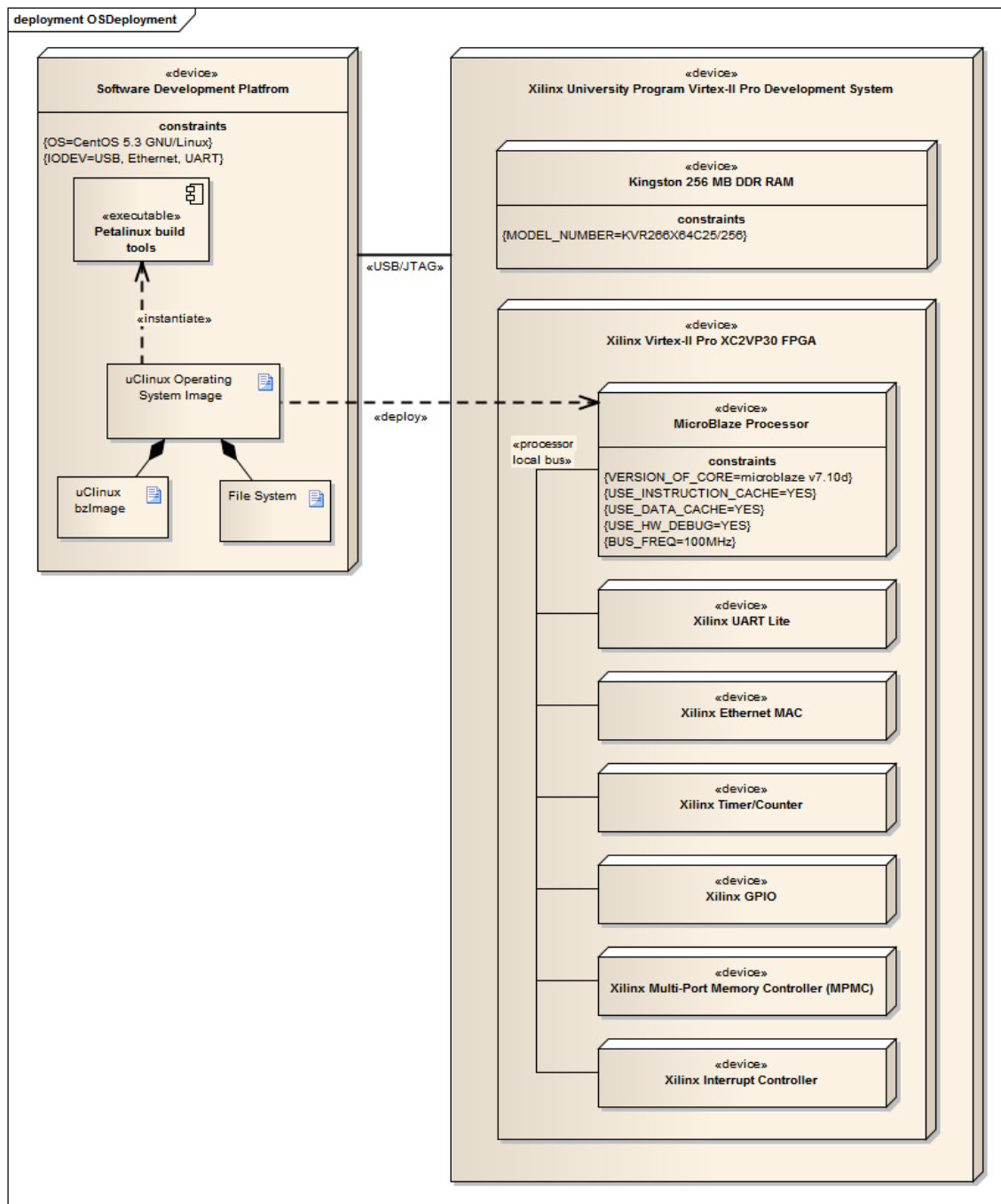
Slika A.3: Diagram postavitve UML: *Namestitev* bitne slike na programirljivo vezje FPGA preko priključka USB/JTAG in vzpostavitev strojne opreme ciljnega sistema



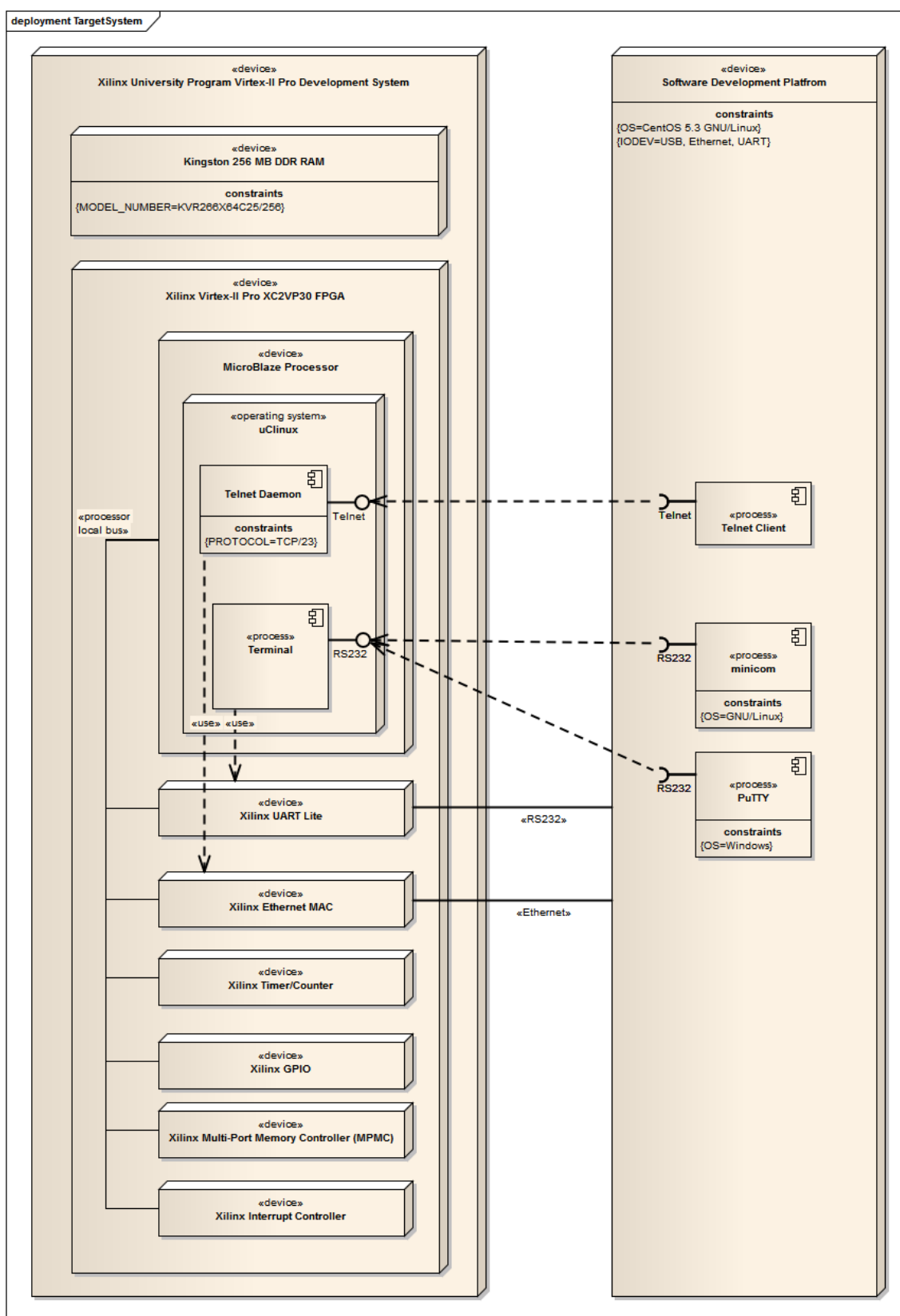
Slika A.4: Diagram postavitve UML: Vzpostavljena strojna oprema ciljnega sistema na programirljivem vezju FPGA



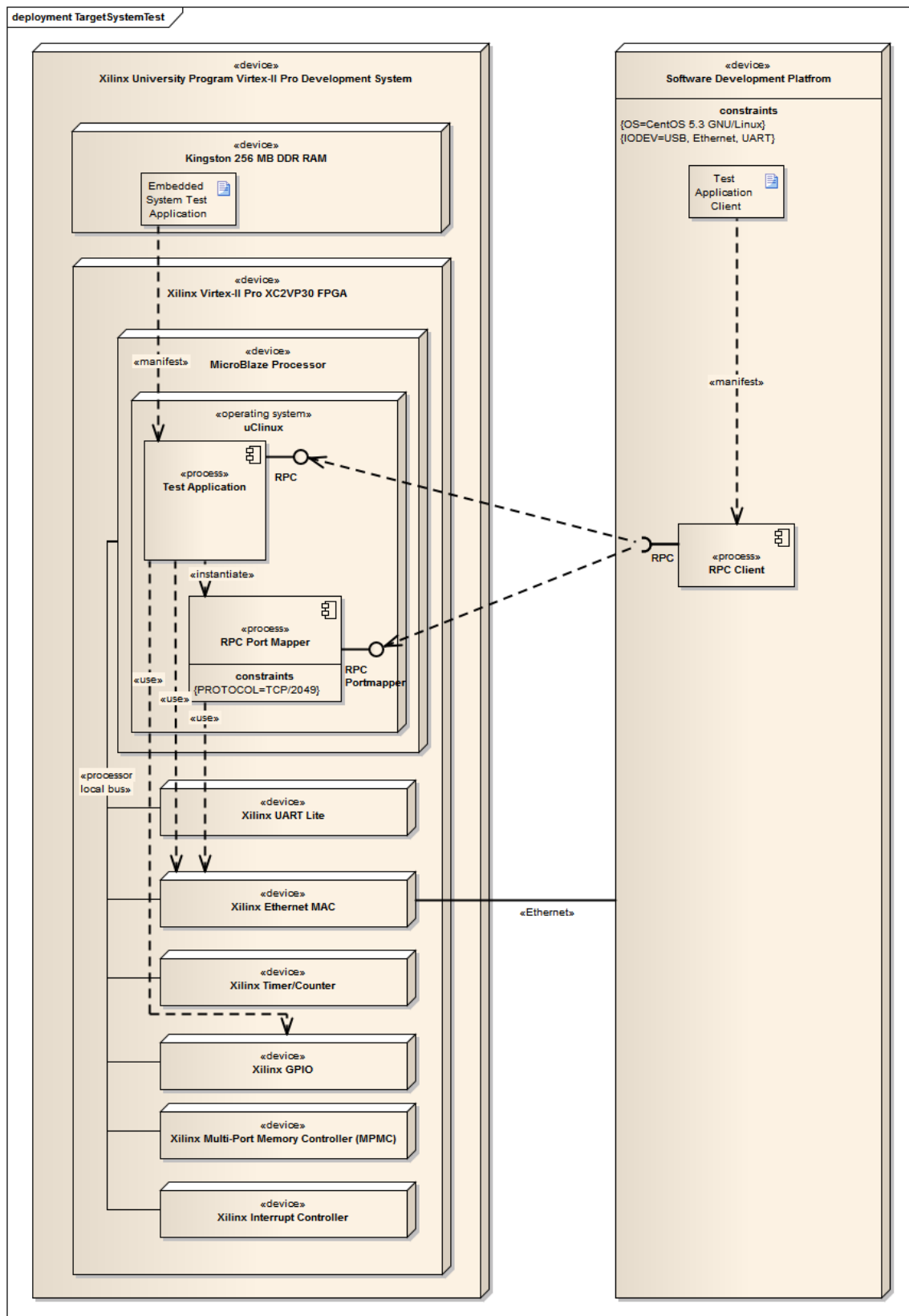
Slika A.5: Diagram postavitve UML: Definicija in gradnja operacijskega sistema uClinux z orodji Petalinux in kreacija datotečnega sistema ter uporabniških programov. Postopek gradnje operacijskega sistema *uporablja* specifikacijo strojne opreme ciljnega sistema in *uporablja* orodja za gradnjo in prevajanje izvirne kode. Ta oprema je bila pridobljena v prvem koraku (glej diagram A.2). Datotečni sistem ciljnega sistema vsebuje dodatno aplikacijo za testiranje omrežne povezljivosti in strojne opreme ciljnega sistema



Slika A.6: Diagram postavitve UML: *Namestitev* slike celotne programske opreme na ciljni sistem preko priključka USB/JTAG z uporabo razhroščevalne enote procesorja MicroBlaze.



Slika A.7: Diagram postavitve UML: Nameščen in delujoč ciljni sistem. Prijava v ciljni sistem je sprva mogoča preko serijskega terminala z uporabo programa PuTTY ali minicom. Če je mrežna naprava ciljnega sistema nastavljena pravilno, se je mogoče prijaviti tudi z oddaljenim dostopom z uporabo odjemalca Telnet



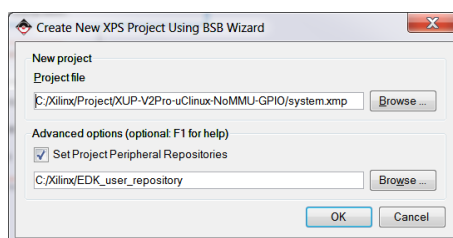
Slika A.8: Diagram postavitve UML: Test oddaljene aplikacije

Postopek načrtovanja, razvoja in namestitve sistema MicroBlaze

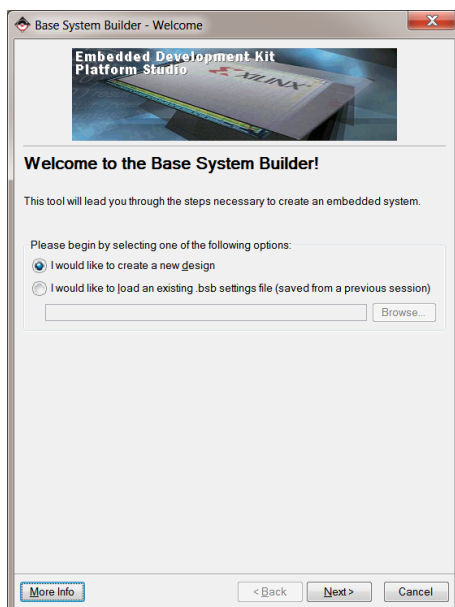
Postopek načrtovanja, razvoja in postavitve strojne opreme in postopek razvoja osnovnih programskih orodij poteka na *Platformi za razvoj Strojne Opreme* na operacijskem sistemu *Windows 7*.

B.1 Postopek definicije strojne opreme

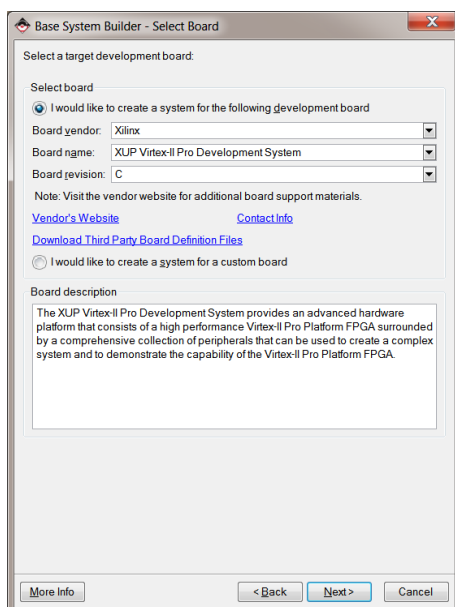
Opomba: Pred samim začetkom postopka definicije strojne opreme je za razvojni sistem Xilinx University Program Virtex-II Pro Development System in EDK 10.0.03 potrebno s spletnega naslova <http://www.xilinx.com/univ/xupv2p.html> naložiti poseben paket *EDK XUPV2P Pack* velikosti 8kB in ga razpakirati v mapo `C:\Xilinx\EDK_user_repsit`. Ta paket vsebuje definicije dodatnih strojnih enot in strojno specifikacijo zgoraj omenjenega razvojnega okolja.



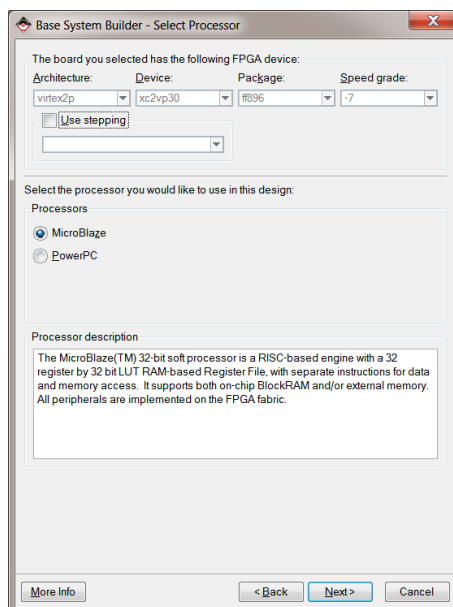
Slika B.1: Kreacija novega projekta s pomočjo čarovnika v programskem orodju EDK in priključitev repozitorija strojne opreme.



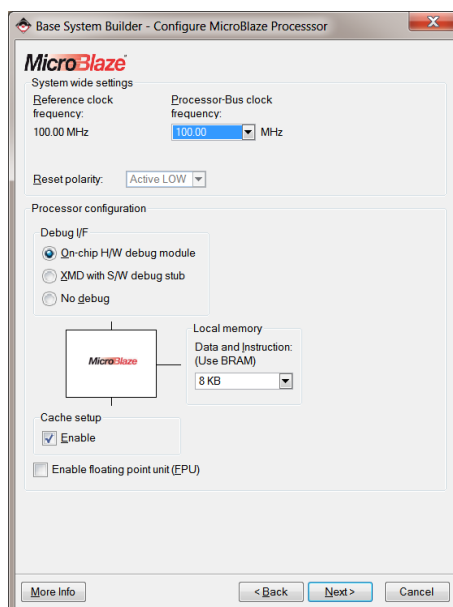
Slika B.2: Kreacija novega načrta.



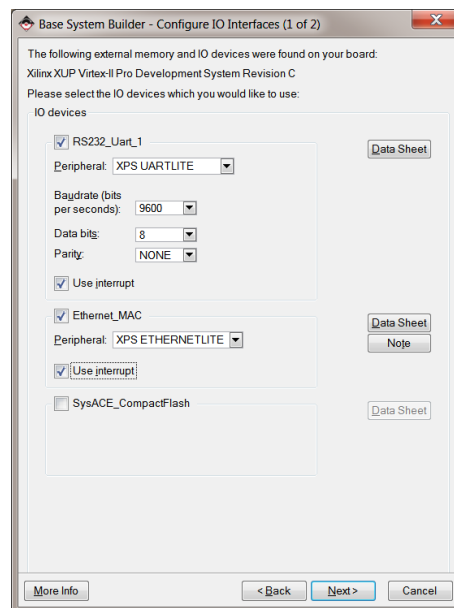
Slika B.3: Določitev razvojnega sistema (za XUP Virtex-II Pro Dev System se zahteva izpolnjen prvi korak - uspešna priključitev repozitorija s slike B.1).



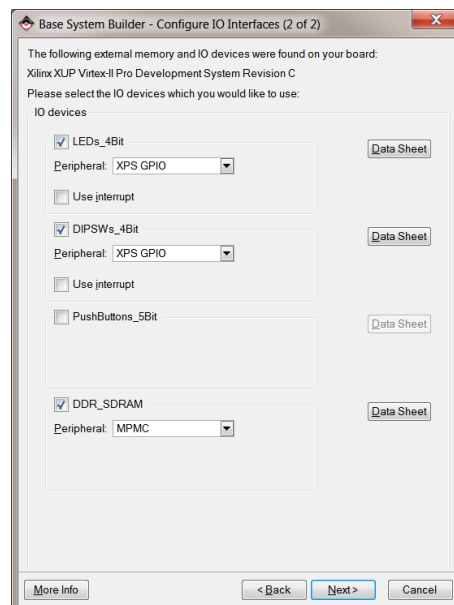
Slika B.4: Izbira programirljivega procesorja MicroBlaze.



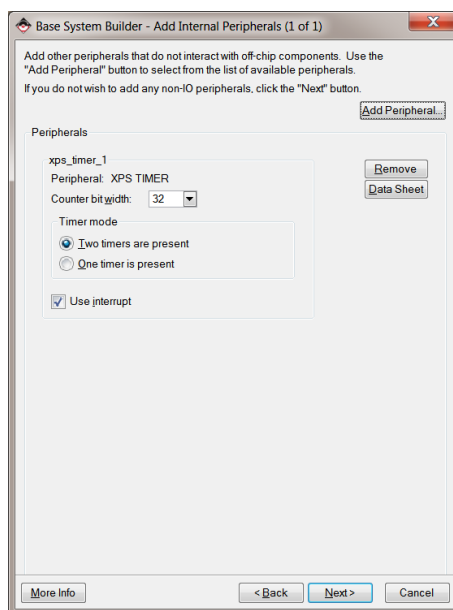
Slika B.5: Določitev osnovnih programirljivih parametrov za kreacijo procesorja MicroBlaze.



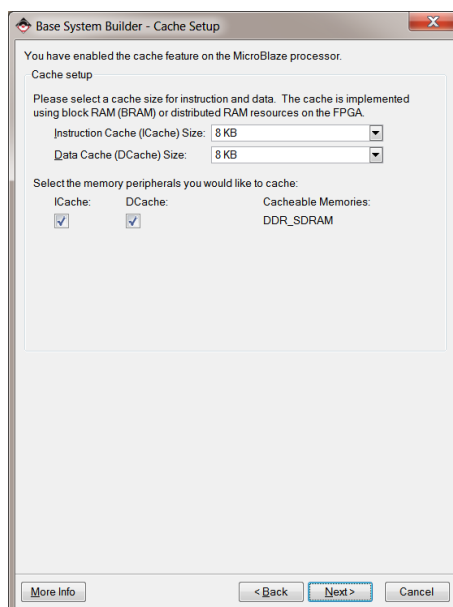
Slika B.6: Izbira in nastavitve dodatnih vhodno/izhodnih naprav (1).



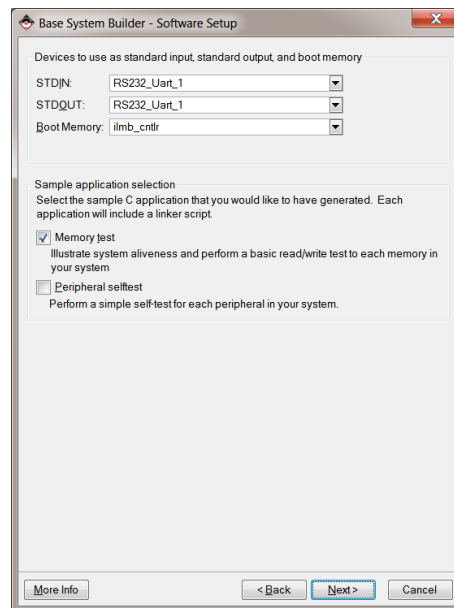
Slika B.7: Izbira in nastavitve dodatnih vhodno/izhodnih naprav (2).



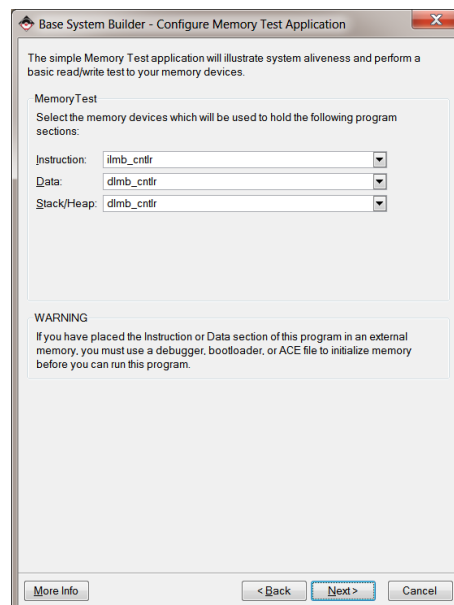
Slika B.8: Izbira in nastavitve dodatnih vhodno/izhodnih naprav (3).



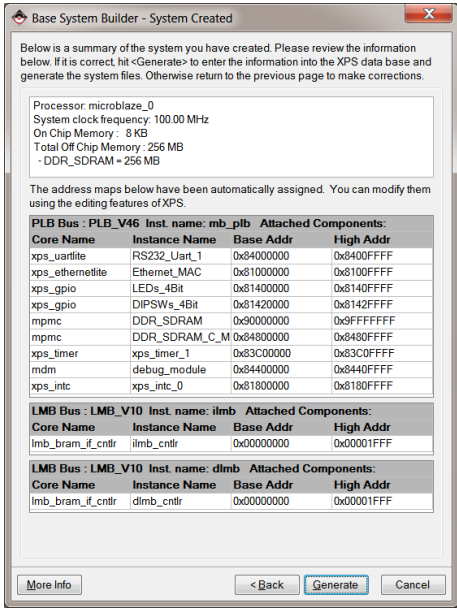
Slika B.9: Nastavitve predpomnilnika procesorja MicroBlaze.



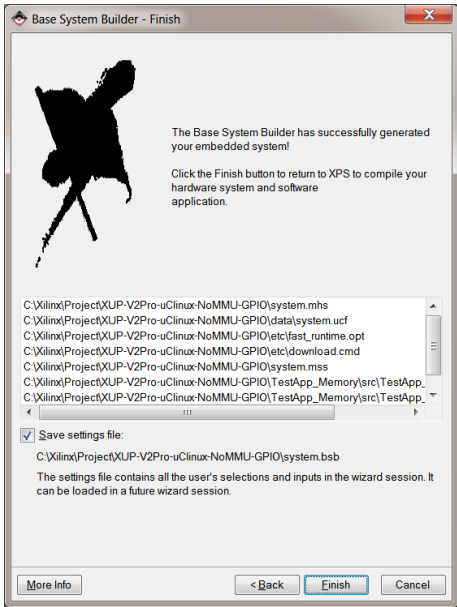
Slika B.10: Izbira privzete vhodno/izhodne naprave, izbira zagonskega pomnilnika in testnih programov.



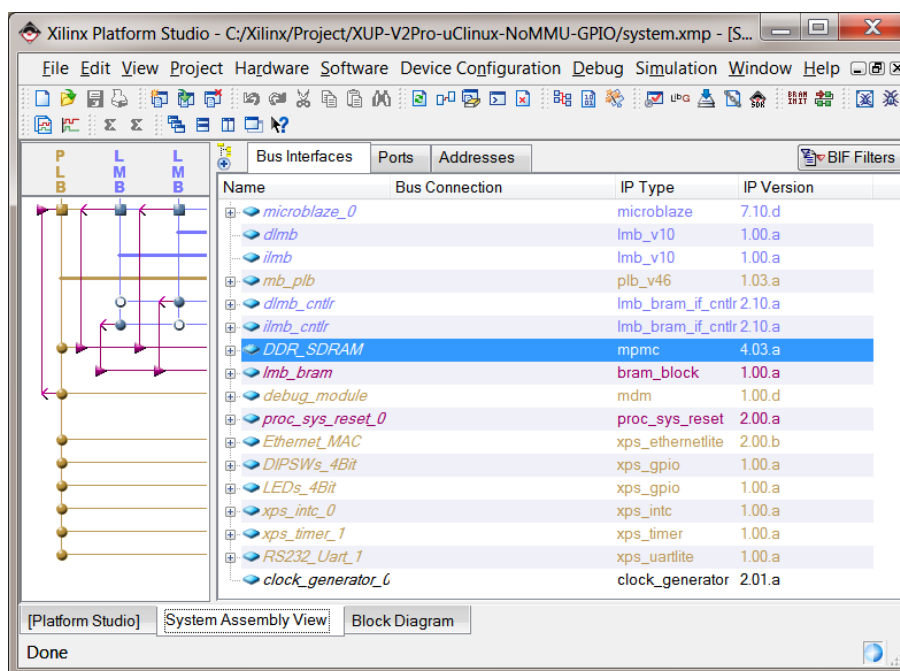
Slika B.11: Določitev parametrov testnega programa.



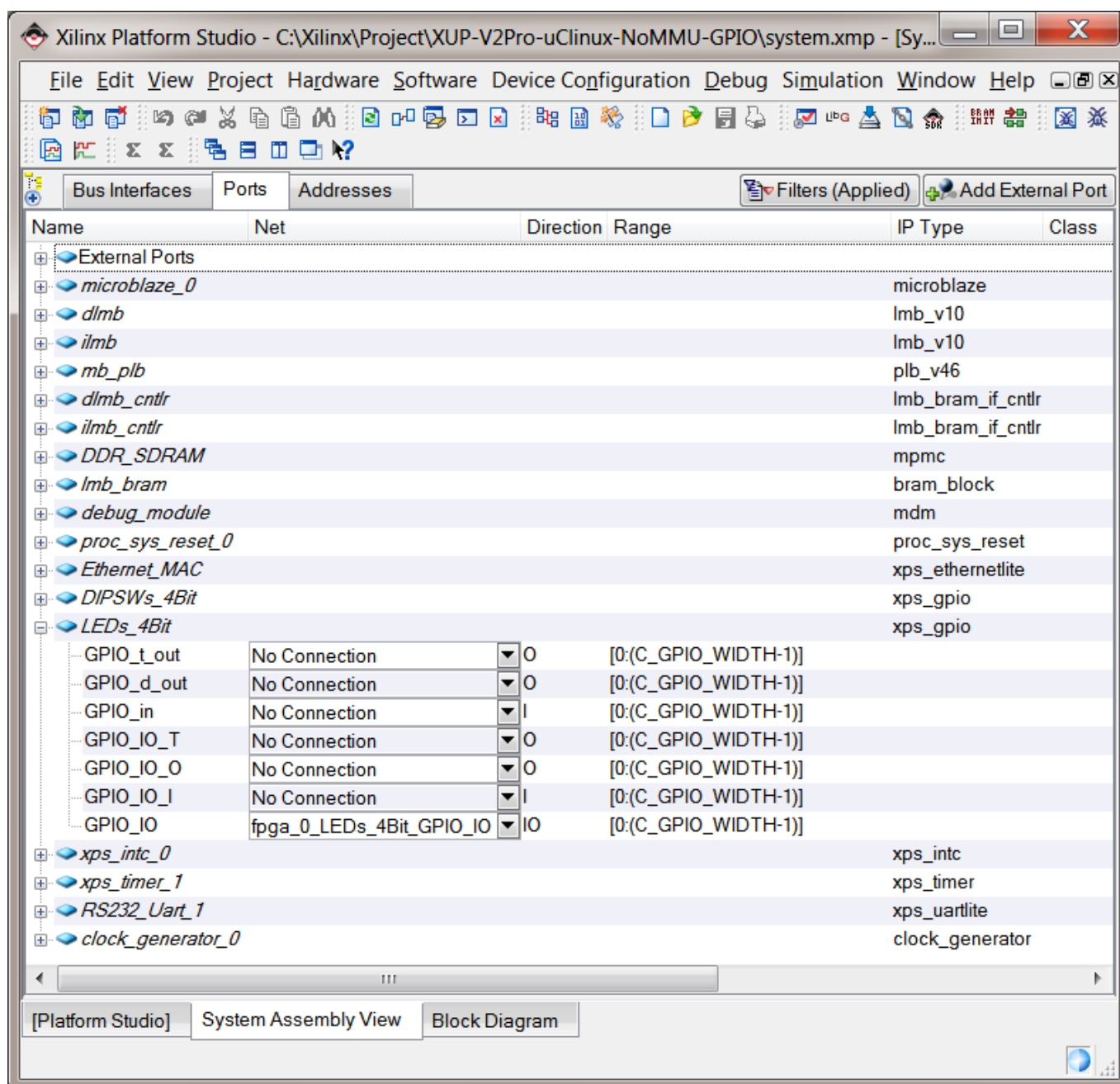
Slika B.12: Povzetek nastavitve strojne opreme.



Slika B.13: Povzetek nastavitve projekta.



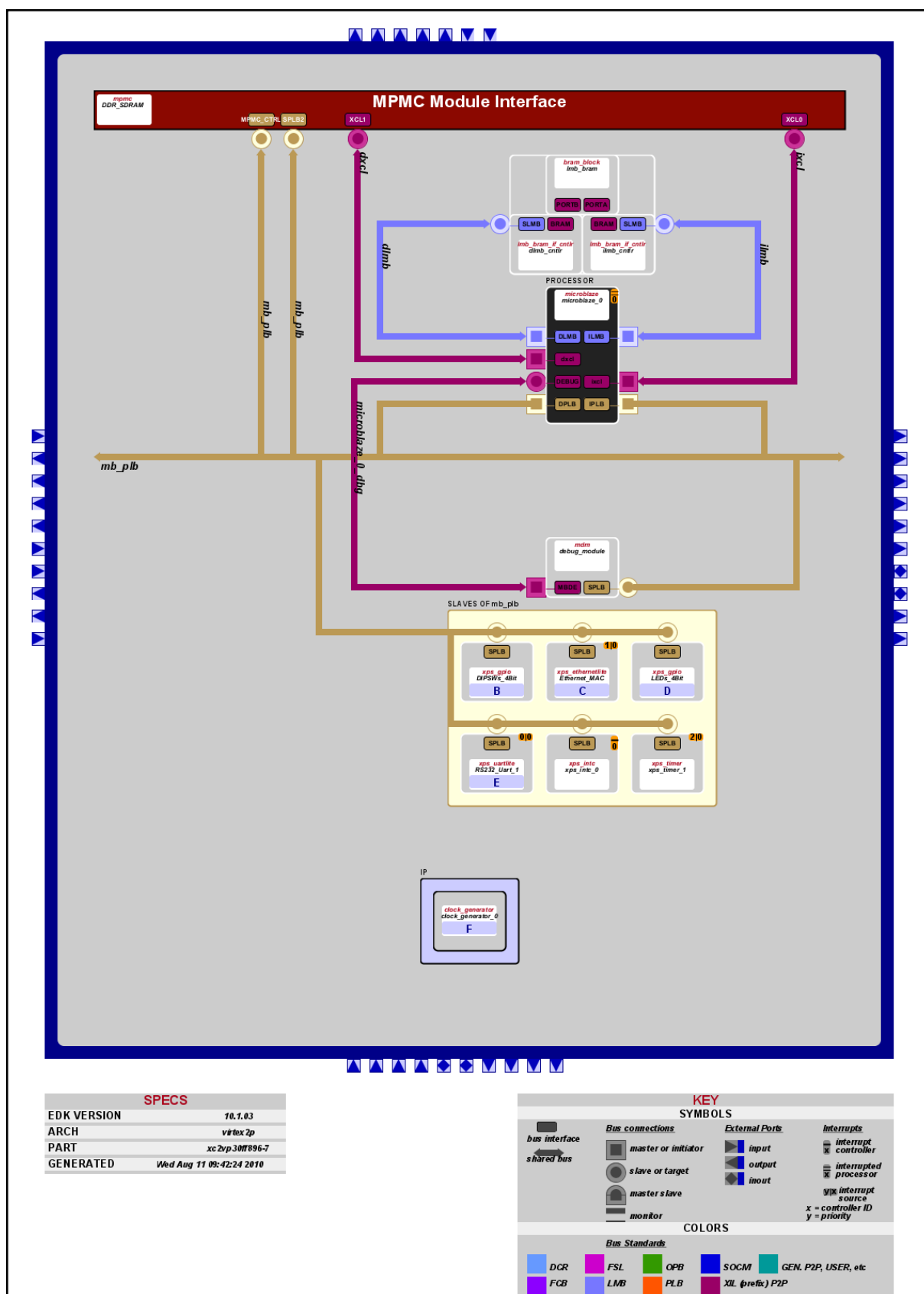
Slika B.14: Projekt: Vidik sestave strojne opreme (1): Določitev vmesnikov vodila računalnika.



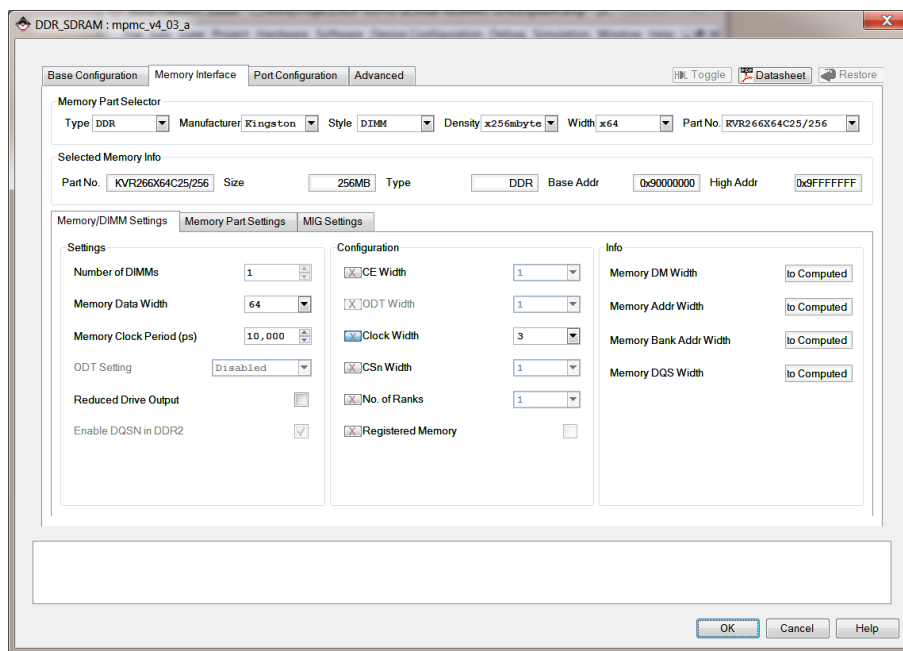
Slika B.15: Projekt: Vidik sestave strojne opreme (2): Priključitev naprav na vhodno/izhodna vrata.

Instance	Name	Base Address	High Address	Size	Bus Interface(s)	Bus Connection	Lock	ICache	DCache	IP Type	IP Version
dlmb_cntlr	C_BASEADDR	0x00000000	0x00001fff	8K	SLMB	dlmb	☒			lmb_bram_if_cntlr	2.10.a
ilmb_cntlr	C_BASEADDR	0x00000000	0x00001fff	8K	SLMB	ilmb	☒			lmb_bram_if_cntlr	2.10.a
debug_module	C_BASEADDR	0x84400000	0x8440ffff	64K	SPLB	mb_plb	☒			mdm	1.00.d
mb_plb	C_BASEADDR			U	Not Applicable		☒			plb_v46	1.03.a
Ethernet_MAC	C_BASEADDR	0x81000000	0x8100ffff	64K	SPLB	mb_plb	☒			xps_ethernetlite	2.00.b
LEDs_4Bit	C_BASEADDR	0x81400000	0x8140ffff	64K	SPLB	mb_plb	☒			xps_gpio	1.00.a
DIPSWs_4Bit	C_BASEADDR	0x81420000	0x8142ffff	64K	SPLB	mb_plb	☒			xps_gpio	1.00.a
xps_intc_0	C_BASEADDR	0x81800000	0x8180ffff	64K	SPLB	mb_plb	☒			xps_intc	1.00.a
xps_timer_1	C_BASEADDR	0x83c00000	0x83c0ffff	64K	SPLB	mb_plb	☒			xps_timer	1.00.a
RS232_Uart_1	C_BASEADDR	0x84000000	0x8400ffff	64K	SPLB	mb_plb	☒			xps_uartlite	1.00.a
DDR_SDRAM	C_MPMC_BASE...	0x90000000	0x9ffffff	256M	XCL0:XCL1:SP...		☒	☒	☒	mpmc	4.03.a
DDR_SDRAM	C_MPMC_CTRL...	0x84800000	0x8480ffff	64K	MPMC_CTRL	mb_plb	☒			mpmc	4.03.a

Slika B.16: Projekt: Vidik sestave strojne opreme (3): Določitev pomnilniških naslovov naprav (*Pomnilnik RAM se nahaja na razponu naslovov 0x90000000 – 0x9FFFFFFF*).

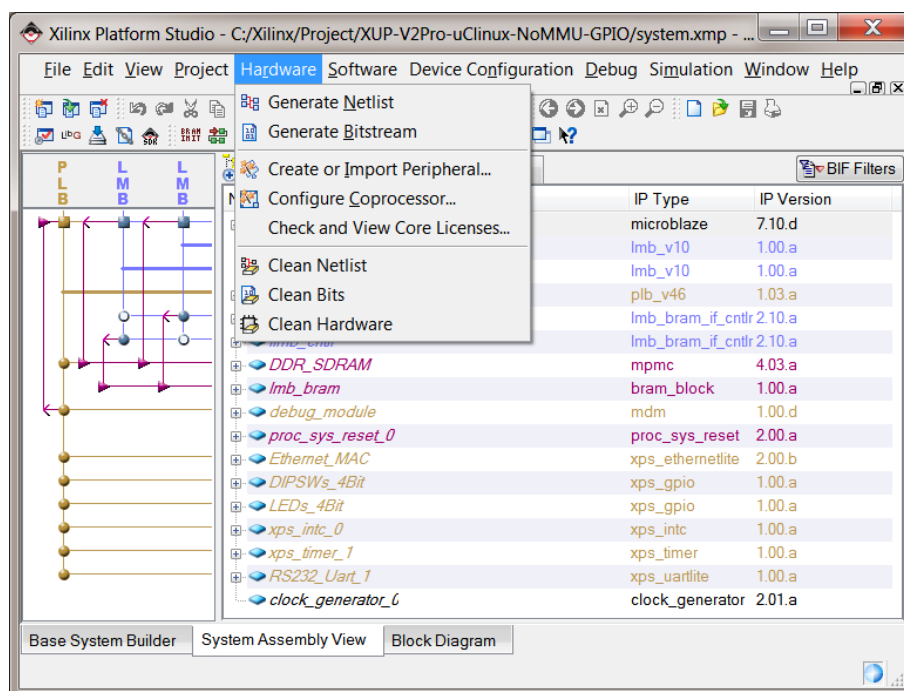


Slika B.17: Projekt: Grafični prikaz organizacije strojne opreme.



Slika B.18: Fina nastavitve krmilnika RAM dostopna s slike B.14 (*Določitev parametrov pomnilnika RAM Kingston KVR266X64C25/256*).

B.2 Postopek gradnje strojne opreme



Slika B.19: Gradnja strojne opreme: Sinteza, implementacija in sestava bitne slike strojne opreme (*Generate Netlist in Generate Bitstream*).

B.3 Postopek gradnje osnovnih programskih orodij

Opomba: Pred samim začetkom postopka definicije programske opreme je potrebno v mapo repozitorija C:\Xilinx\EDK_user_repsitory prekopirati v mapo Petalinux z nastavitvenimi datotekami za nastavitev pred gradnjo osnovnih programskih orodij, ki bodo v nadaljevanju uporabljene za gradnjo operacijskega sistema. Mapa se nahaja v arhivu petalinux-v0.40-final.tar.gz na poti hardware/edk_user_repository/Petalogix. Po izvedenem postopku repozitorij EDK vsebuje datoteke spodnjega izpisa.

```

1 11.08.2010  20:41    <DIR>      .
2 11.08.2010  20:41    <DIR>      ..
3 28.01.2010  12:52    <DIR>      PetaLogix
4 28.01.2010  12:51    <DIR>      XilinxProcessorIP
5                      0 File(s)      0 bytes
6
7 Directory of C:\Xilinx\EDK_user_repository\PetaLogix
8
9 28.01.2010  12:52    <DIR>      .
10 28.01.2010  12:52    <DIR>      ..
11 28.01.2010  12:52    <DIR>      bsp
12                      0 File(s)      0 bytes
13
14 Directory of C:\Xilinx\EDK_user_repository\PetaLogix\bsp
15
16 28.01.2010  12:52    <DIR>      .
17 28.01.2010  12:52    <DIR>      ..
18 28.01.2010  12:52    <DIR>      petalinux_v1_00_a
19 28.01.2010  12:52    <DIR>      petalinux_v1_00_b
20                      0 File(s)      0 bytes
21
22 Directory of C:\Xilinx\EDK_user_repository\PetaLogix\bsp\petalinux_v1_00_a
23
24 28.01.2010  12:52    <DIR>      .
25 28.01.2010  12:52    <DIR>      ..
26 28.01.2010  12:52    <DIR>      data
27                      0 File(s)      0 bytes
28
29 Directory of C:\Xilinx\EDK_user_repository\PetaLogix\bsp\petalinux_v1_00_a\data
30
31 28.01.2010  12:52    <DIR>      .
32 28.01.2010  12:52    <DIR>      ..
33 28.01.2010  12:12          2.983 petalinux_v2_1_0.mld
34 28.01.2010  12:12      22.910 petalinux_v2_1_0.tcl
35                      2 File(s)      25.893 bytes
36
37 Directory of C:\Xilinx\EDK_user_repository\PetaLogix\bsp\petalinux_v1_00_b
38
39 28.01.2010  12:52    <DIR>      .
40 28.01.2010  12:52    <DIR>      ..
41 28.01.2010  12:52    <DIR>      data
42                      0 File(s)      0 bytes
43
44 Directory of C:\Xilinx\EDK_user_repository\PetaLogix\bsp\petalinux_v1_00_b\data
45
46 28.01.2010  12:52    <DIR>      .
47 28.01.2010  12:52    <DIR>      ..
48 28.01.2010  12:12          3.870 petalinux_v2_1_0.mld
49 28.01.2010  12:12      46.082 petalinux_v2_1_0.tcl
50                      2 File(s)      49.952 bytes
51
52 Directory of C:\Xilinx\EDK_user_repository\XilinxProcessorIP
53
54 28.01.2010  12:51    <DIR>      .
55 28.01.2010  12:51    <DIR>      ..

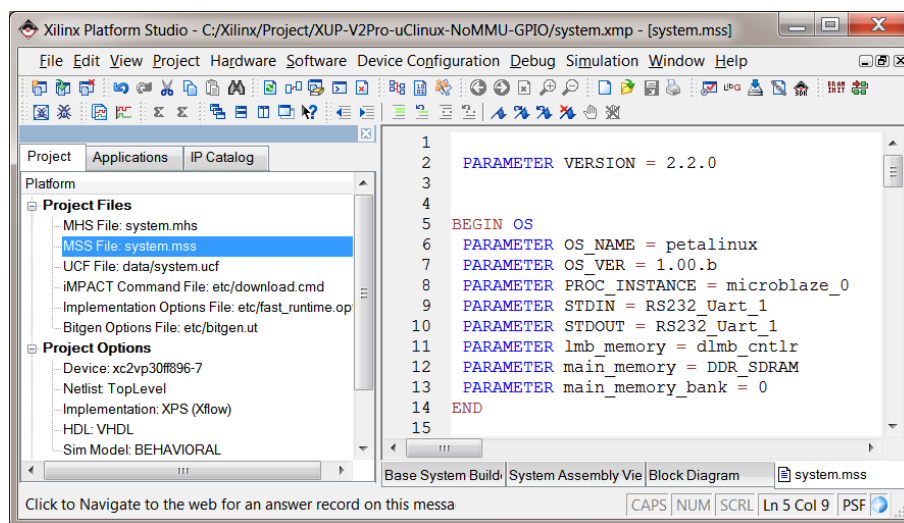
```

```

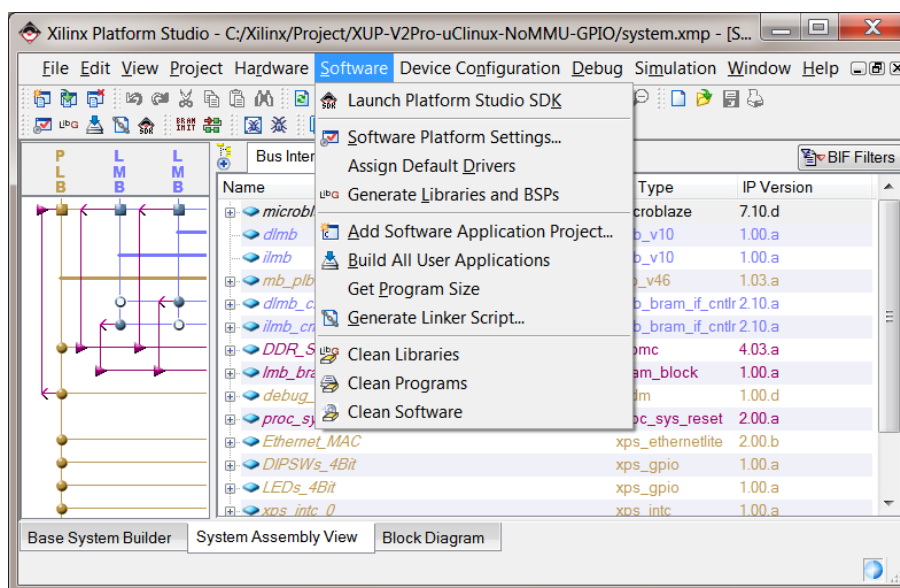
56 28.01.2010 12:51 <DIR> boards
57 0 File(s) 0 bytes
58
59 Directory of C:\Xilinx\EDK_user_repository\XilinxProcessorIP\boards
60
61 28.01.2010 12:51 <DIR> .
62 28.01.2010 12:51 <DIR> ..
63 28.01.2010 12:51 <DIR> Xilinx_XUP_V2P
64 0 File(s) 0 bytes
65
66 Directory of C:\Xilinx\EDK_user_repository\XilinxProcessorIP\boards\Xilinx_XUP_V2P
67
68 28.01.2010 12:51 <DIR> .
69 28.01.2010 12:51 <DIR> ..
70 28.01.2010 12:51 <DIR> data
71 0 File(s) 0 bytes
72
73 Directory of C:\Xilinx\EDK_user_repository\XilinxProcessorIP\boards\Xilinx_XUP_V2P\data
74
75 28.01.2010 12:51 <DIR> .
76 28.01.2010 12:51 <DIR> ..
77 07.05.2008 13:37 105 DDR_SDRAM_mpmc.ucf
78 05.08.2008 12:02 38.077 Xilinx_XUP_V2P_v2_2_0.xbd
79 2 File(s) 38.182 bytes
80
81 Total Files Listed:
82 7 File(s) 114.027 bytes
83 32 Dir(s)

```

Izpis B.1: UKAZNA LUPINA - Vsebina repozitorija.

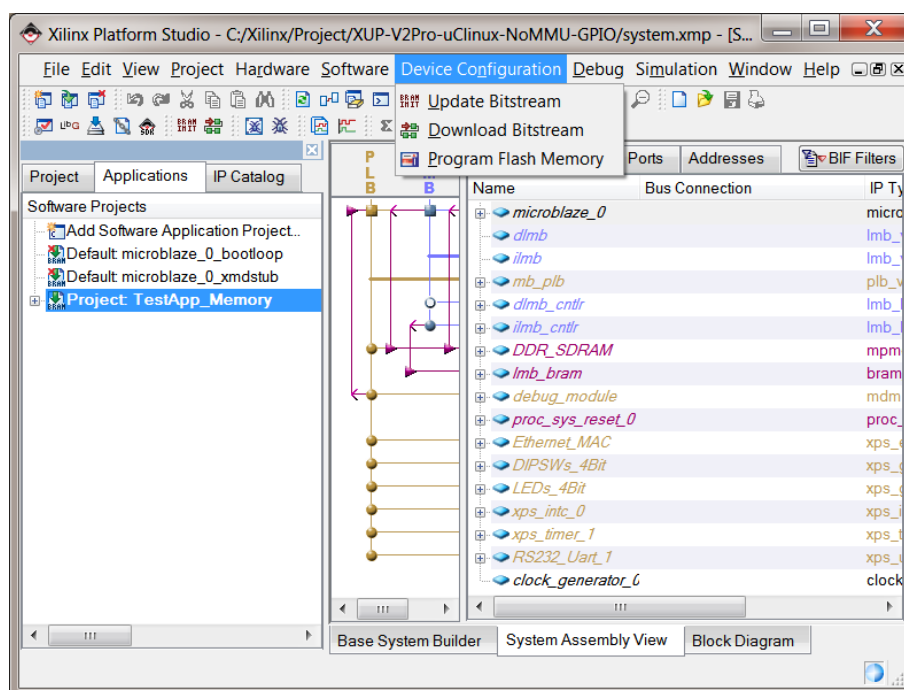


Slika B.20: Sprememba vnosa datoteke *system.mss*: Sprememba definicije operacijskega sistema (*vnos je mogoče spremeniti tudi preko grafičnega vmesnika v Software:Software Platform Settings:Software Platform in Software:Software Platform Settings:OS and Libraries*).

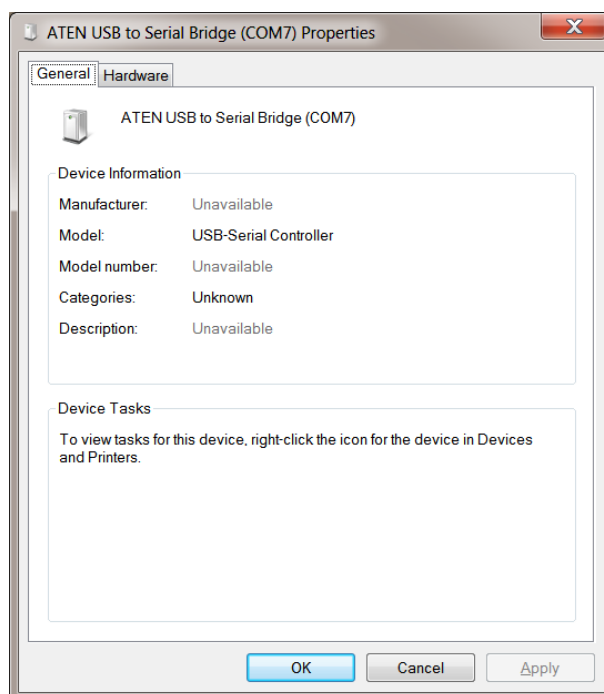


Slika B.21: Gradnja osnovne programske opreme z lastnimi določili za operacijski sistem (BSP): Gradnja knjižnic in vseh uporabniških aplikacij (*Generate Libraries and BSPs*, *Build All User Application*).

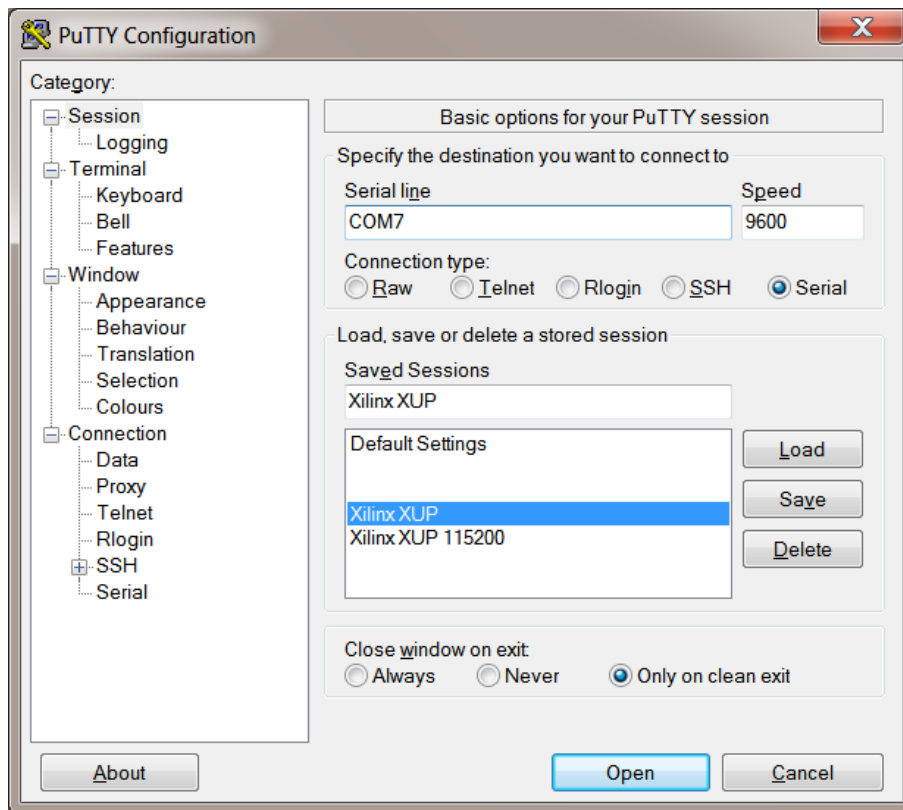
B.4 Vmesni preizkus strojne opreme



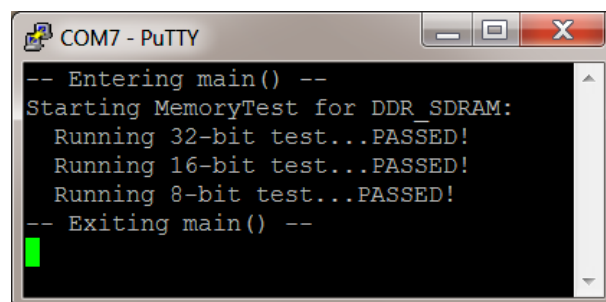
Slika B.22: Pred nadaljevanjem se je najbolje prepričati ali se je postopek gradnje strojne opreme in programskih orodij pravilno zaključil. Za ta namen se posodobi bitna slika strojne in programske opreme (*Update Bitstream*) ter namesti na sistem FPGA z uporabo povezave USB/JTAG (*Download bitstream*).



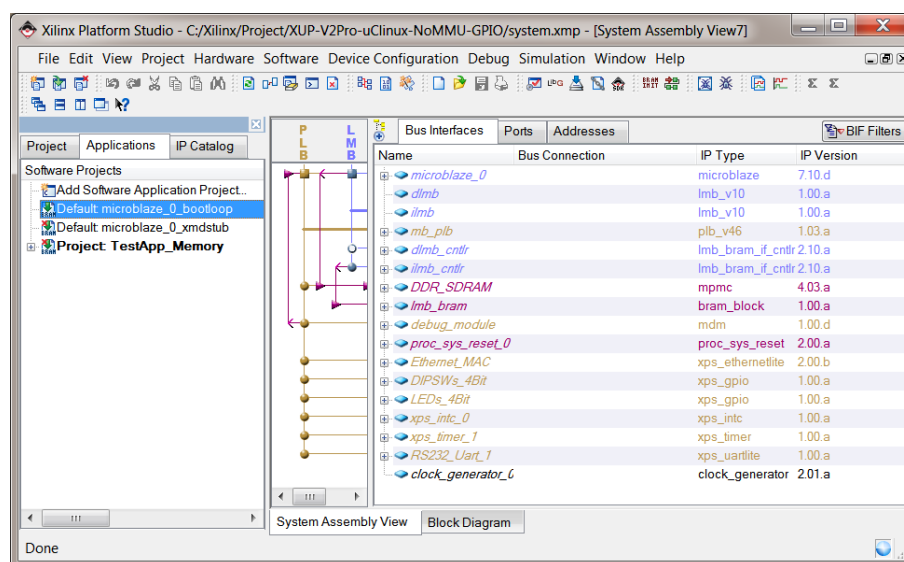
Slika B.23: V programskih orodjih je bil določen standardni vhod/izhod UART, zato se za dostop do testnega sistema uporabi serijsko vodilo (*v našem primeru je dostop posreden na vratih COM7 z uporabo adapterja USB/RS232 Aten UC-232A*).



Slika B.24: Za dostop se uporabi odjemalec PuTTY.



Slika B.25: Z malo sreče ☺ se na serijskem terminalu izpiše besedilo uspešnega testiranja pomnilnika DDR RAM (*sicer poskusite ponovno ponastaviti sistem (RESET)*). To je tudi eden izmed *potrebnih pogojev* za uspešno namestitve slike operacijskega sistema in datotečne sistema, ki se bosta nahajala v pomnilniku RAM.



Slika B.26: Zadnji korak je sprememba izbire aktivnega programa za neskončno zanko. Operacijski sistem se bo v nadaljevanju naložil preko razhroščevalnega vmesnika MicroBlaze z uporabo povezave USB/JTAG.

B.5 Povzetek poglavja

Postopki diagramov A.2, A.3 in A.4 s poglavja A so zaključeni in predelani. Strojna oprema je definirana, zgrajena in nameščena na programirljivo vezje FPGA.

V tem poglavju so se prav tako zgradila osnovna programska orodja za gradnjo in prevažanje programske kode, nekatere sistemske knjižnice in gonilniki. Ta orodja bodo v nadaljevanju uporabljena pri grajenju operacijskega sistema uClinux.

Definicija strojne naprave GPIO je podana v podatkovnem listu.

Postopek nalaganja strojne opreme je potrebno opraviti po vsaki ponastavitvi programirljivega vezja. Postopek se lahko opusti le v primeru, če se bitna slika nahaja v pomnilniku FLASH razvojnega sistema in je samodejna nastavitvev FPGA s pomnilnika FLASH omogočena.

Postopek načrtovanja, razvoja in namestitve operacijskega sistema Petalinux

Postopek načrtovanja, razvoja in namestitve operacijskega sistema Petalinux za ciljni sistem poteka na *Platformi za razvoj Programske Opreme* z operacijskim sistemom *CentOS 5.3 GNU/Linux*.

Postopek namestitve slike operacijskega in datotečnega sistema poteka na *Platformi za razvoj Strojne Opreme* z operacijskim sistemom *Windows 7*.

C.1 Priprava okolja za definicijo in gradnjo operacijskega sistema uClinux

```
1 [root@devpltfrm ~]# mkdir uclinux
2 [root@devpltfrm ~]# tar xzf /mnt/petalinux-v0.40-final.tar.gz -C /root/uclinux
3 [root@devpltfrm ~]# vim appbuild-env.sh
4 [root@devpltfrm ~]# cat appbuild-env.sh
5 #!/bin/bash
6
7 # PetaLinux env settings
8 cd $(pwd)/petalinux-v0.40-final
9 source settings.sh 1> /dev/null
10
11 # MicroBlaze-uClinux build tools PATH include
12
13 PATH=${PETALINUX}/tools/linux-i386/microblaze-uclinux-tools/bin:${PATH}
14
15 # Change dir
16
17 cd ${PETALINUX}/software
18
19 echo 'PetaLinux Environment'
20 bash
21 #EOF
22
23 [root@devpltfrm ~]# chmod +x appbuild-env.sh
24 [root@devpltfrm ~]# ./appbuild-env.sh
```

136 Poglavje C: Postopek načrtovanja, razvoja in namestitve operacijskega sistema Petalinux

25 PetaLinux Environment

26 [**r**oot@devpltfrm software]#

Izpis C.1: UKAZNA LUPINA - Razpakiranje arhiva uClinux in orodij Petalinux ter kreacija skripte za hitri dostop do razvojnega okolja Petalinux.

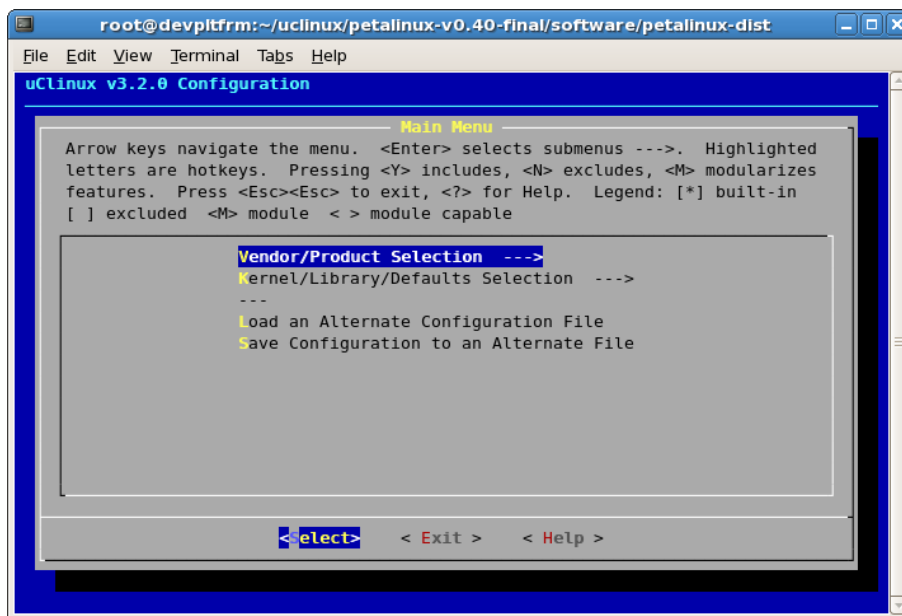
```
1 [root@devpltfrm ~]# cp -r /mnt/XUP-V2Pro-uClinux-NoMMU-GPIO →  
    ↪ /root/uclinux/petalinux-v0.40-final/hardware/user-platforms #Izvoz projekta EDK v →  
    ↪ mapo uporabniških platform Petalinux  
2  
3 [root@devpltfrm ~]# petalinux-new-platform -v Xilinx -p XUP-V2Pro-uClinux-NoMMU-GPIO -k 2.6 →  
    ↪ #Deklaracija nove platforme uvoženega projekta za različico kernela uClinux 2.6  
4  
5 New platform for Xilinx XUP-V2Pro-uClinux-NoMMU-GPIO successfully created  
6 Please use petalinux-copy-autoconfig tools with the same parameters (-v, -p)
```

Izpis C.2: UKAZNA LUPINA - Namestitev osnovnih knjižnic za gradnjo.

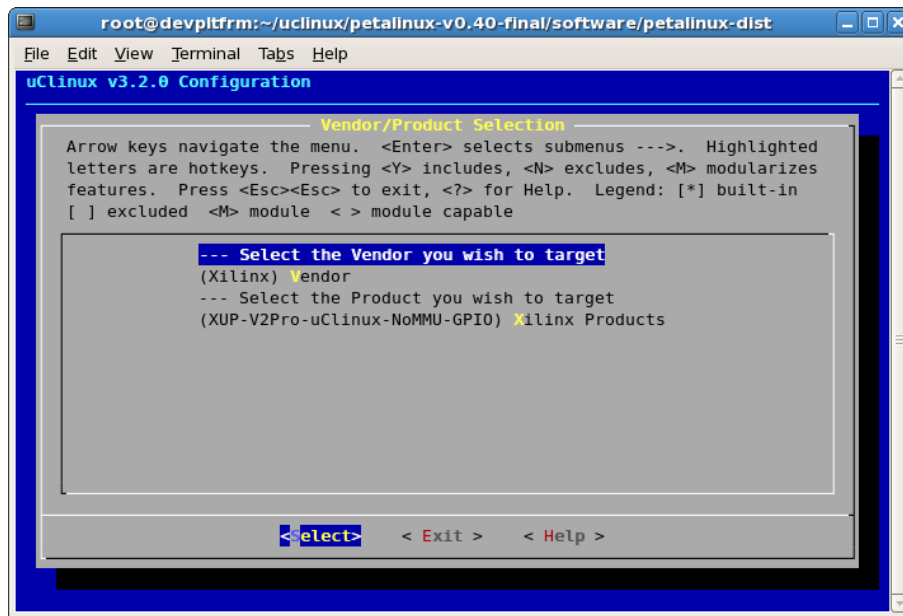
C.2 Postopek sinhronizacije nastavitev strojne opreme in okolja Petalinux

```
1 [root@devpltfm ~]# cd $PETALINUX/software/petalinux-dist
2 [root@devpltfm petalinux-dist]# make menuconfig
```

Izpis C.3: UKAZNA LUPINA - Izbira strojne platforme in projekta za gradnjo operacijskega sistema.



Slika C.1: Izbira projekta (1).



Slika C.2: Izbira projekta (2).

```

1 [root@devpltfm ~]# cd $PETALINUX/software/petalinux-dist
2 [root@devpltfm petalinux-dist]# cd →
   ↳ $PETALINUX/hardware/user-platforms/XUP-V2Pro-uClinux-NoMMU-GPIO/
3 [root@devpltfm user-platforms]# petalinux-copy-autoconfig #Sinhronizacija nastavitev
4
5 INFO: Attempting vendor/platform auto-detect
6 INFO: Auto-detected Xilinx/XUP-V2Pro-uClinux-NoMMU-GPIO combination.
7 Auto-config file successfully updated for Xilinx XUP-V2Pro-uClinux-NoMMU-GPIO

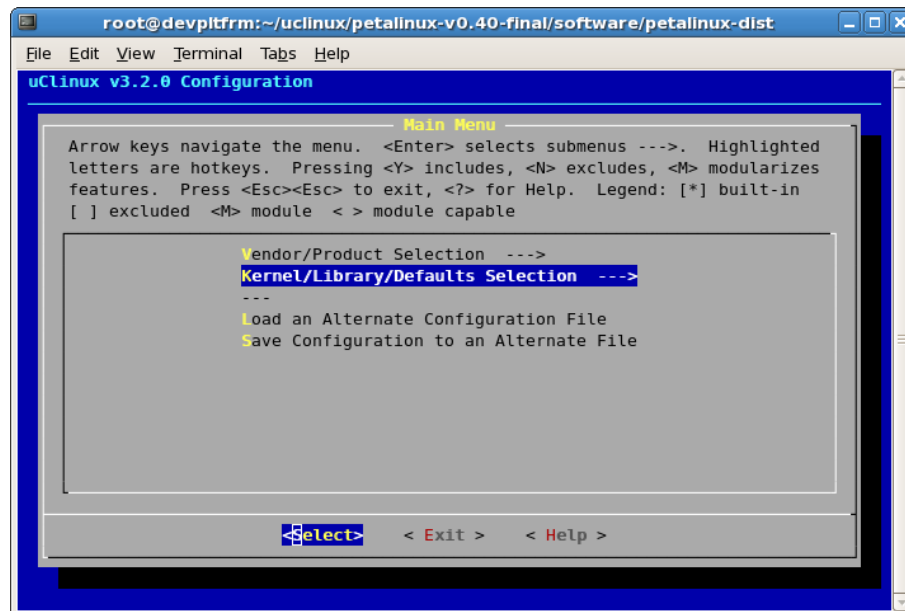
```

Izpis C.4: UKAZNA LUPINA - Izbira strojne platforme in projekta za gradnjo operacijskega sistema.

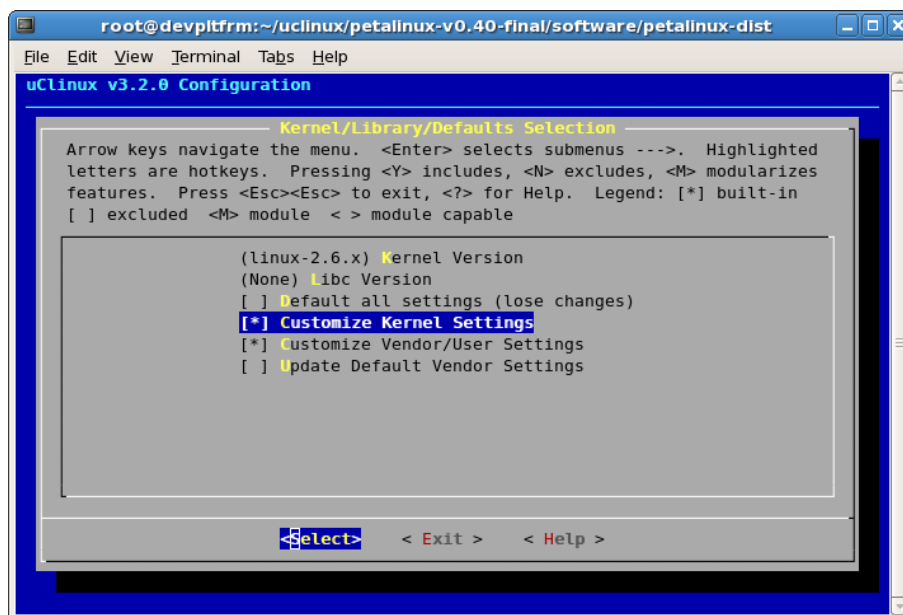
C.3 Gradnja operacijskega sistema

```
1 [root@devpltfrm user-platforms]# cd $PETALINUX/software/petalinux-dist
2 [root@devpltfrm petalinux-dist]# make menuconfig
```

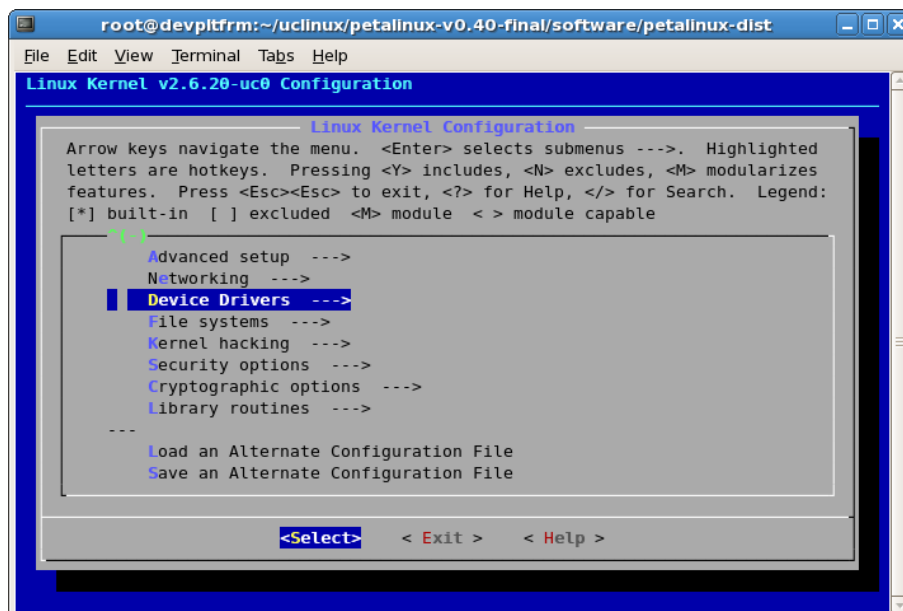
Izpis C.5: UKAZNA LUPINA - Izbira strojne platforme in projekta za gradnjo operacijskega sistema.



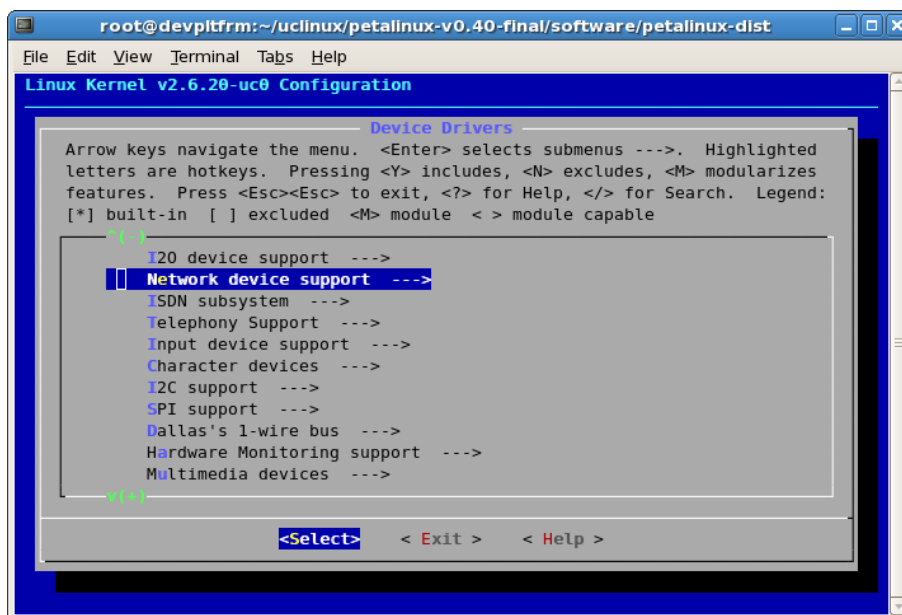
Slika C.3: Izbira nastavitev jedra operacijskega sistema (1).



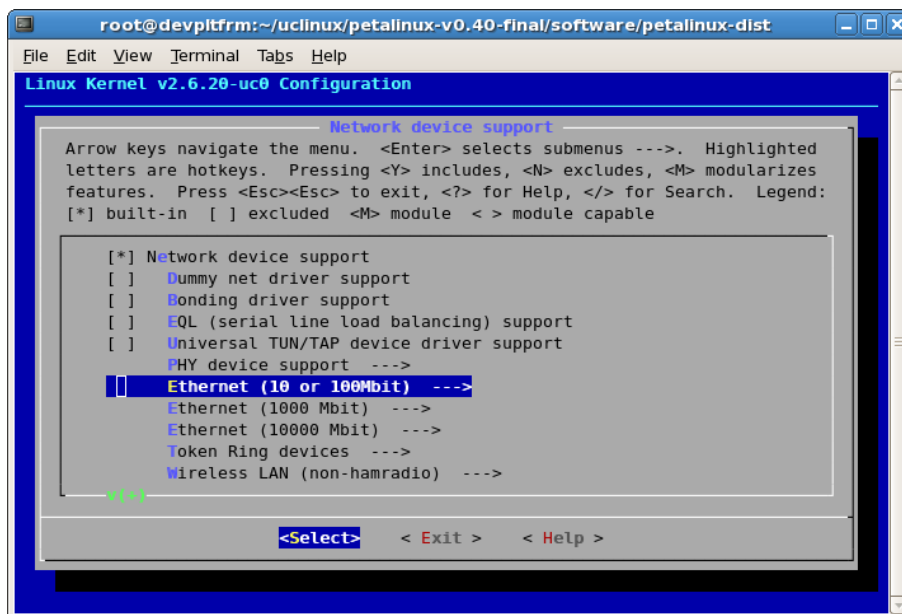
Slika C.4: Izbira nastavitev jedra operacijskega sistema (2).



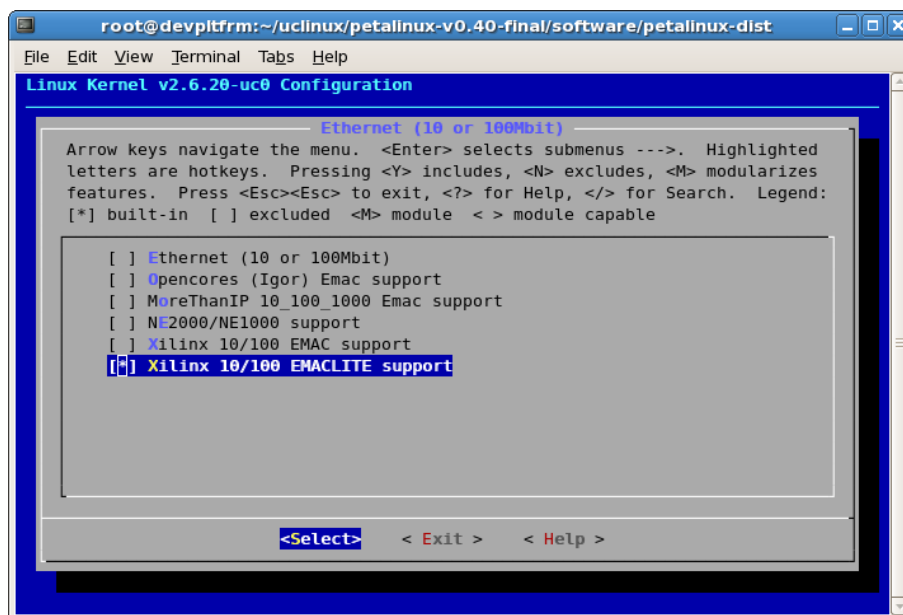
Slika C.5: Izbira nastavitev jedra operacijskega sistema (3).



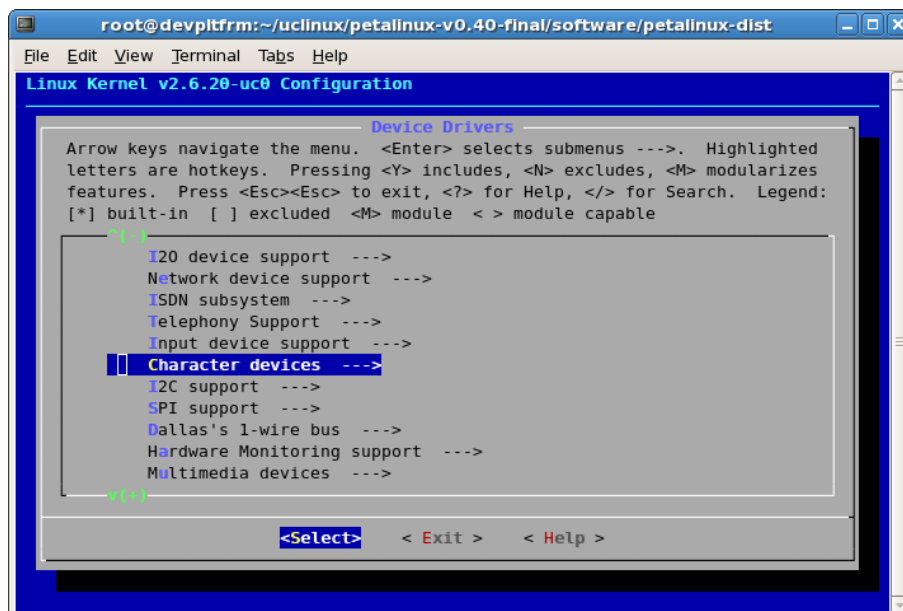
Slika C.6: Izbira nastavitev jedra operacijskega sistema (4).



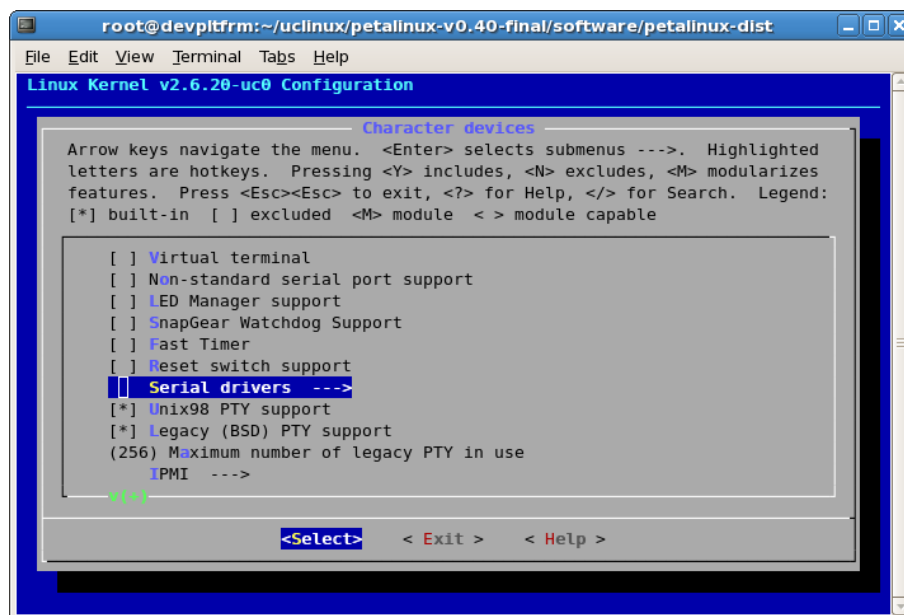
Slika C.7: Izbira nastavitev jedra operacijskega sistema (5).



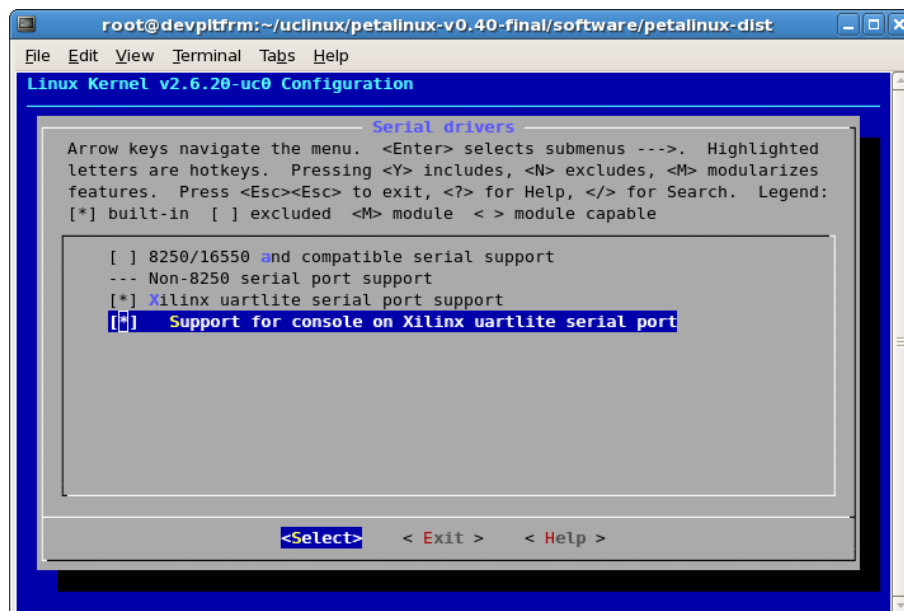
Slika C.8: Izbira nastavitev jedra operacijskega sistema (6).



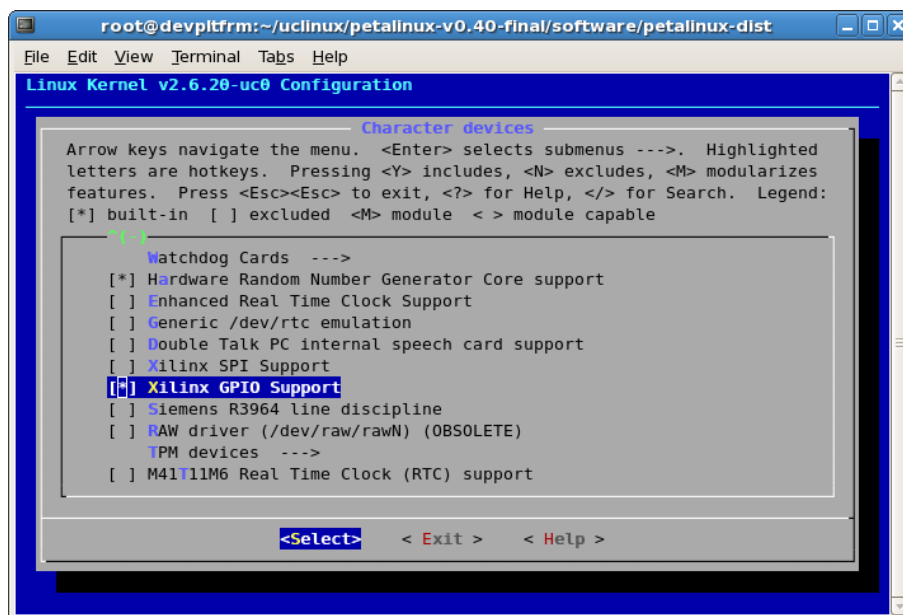
Slika C.9: Izbira nastavitev jedra operacijskega sistema (7).



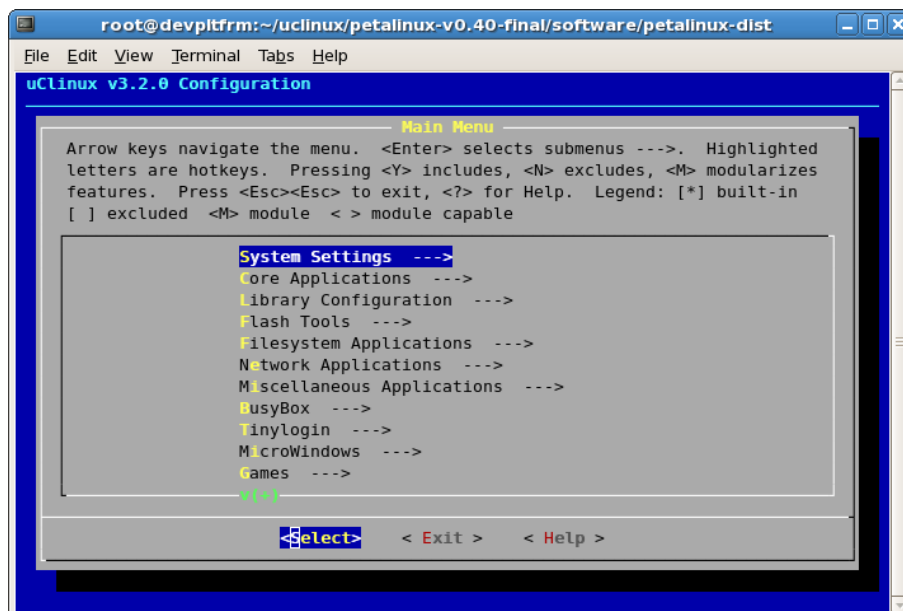
Slika C.10: Izbira nastavitev jedra operacijskega sistema (8).



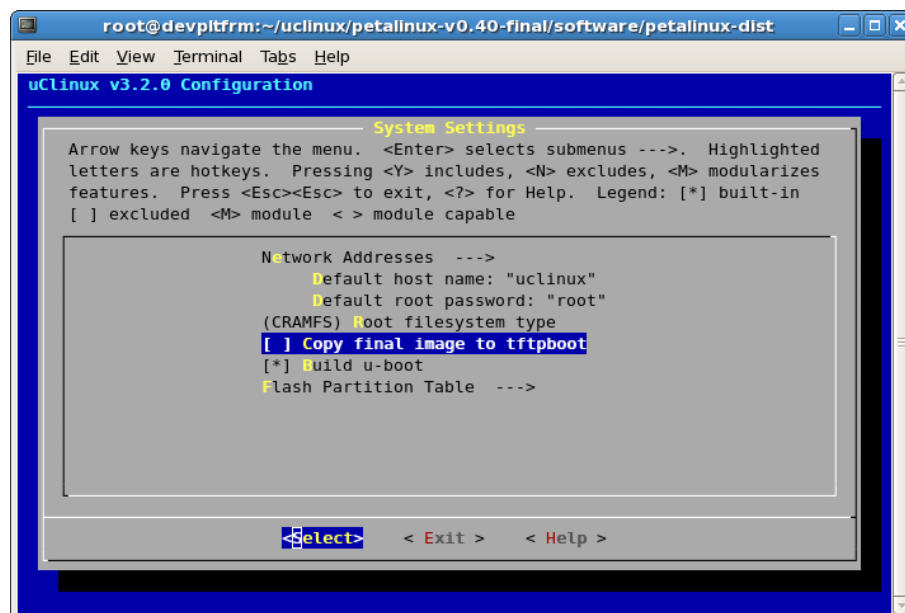
Slika C.11: Izbira nastavitev jedra operacijskega sistema (9).



Slika C.12: Izbira nastavitev jedra operacijskega sistema (10).



Slika C.13: Izbira nastavitev uporabniške programske opreme (1).



Slika C.14: Izbira nastavitev uporabniške programske opreme (2).

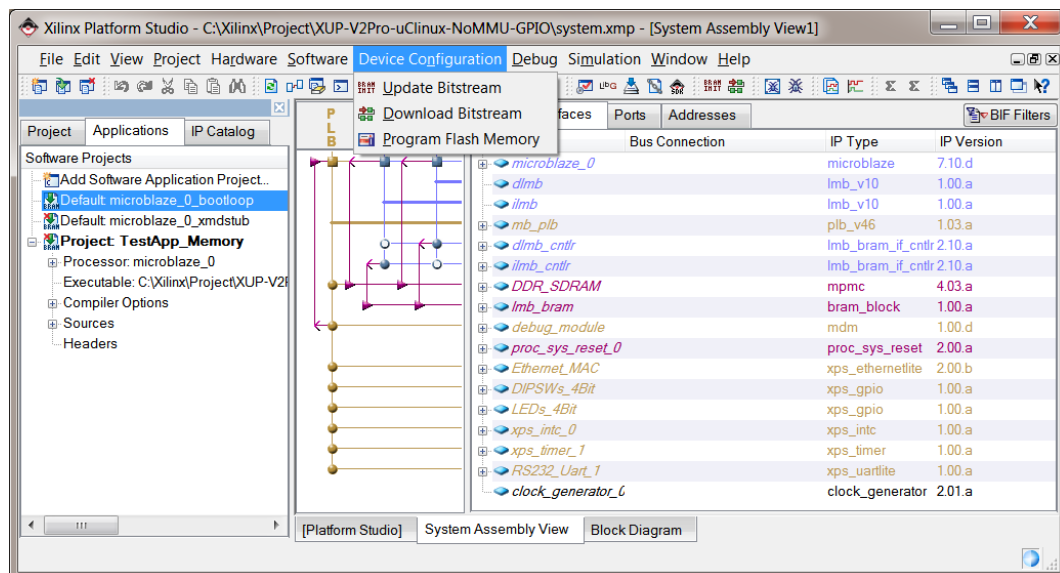
```
1 [root@devpltfrm petalinux-dist]# make
2 [root@devpltfrm petalinux-dist]# ls images
3
4 image.bin    image.ub      rootfs.cpio   u-boot.elf    u-boot.srec
5 image.elf     linux.bin     ub.config.img u-boot-s.bin  u-boot-s.srec
6 image.srec    romfs.img     u-boot.bin    u-boot-s.elf
```

Izpis C.6: UKAZNA LUPINA - Gradnja operacijskega sistema.

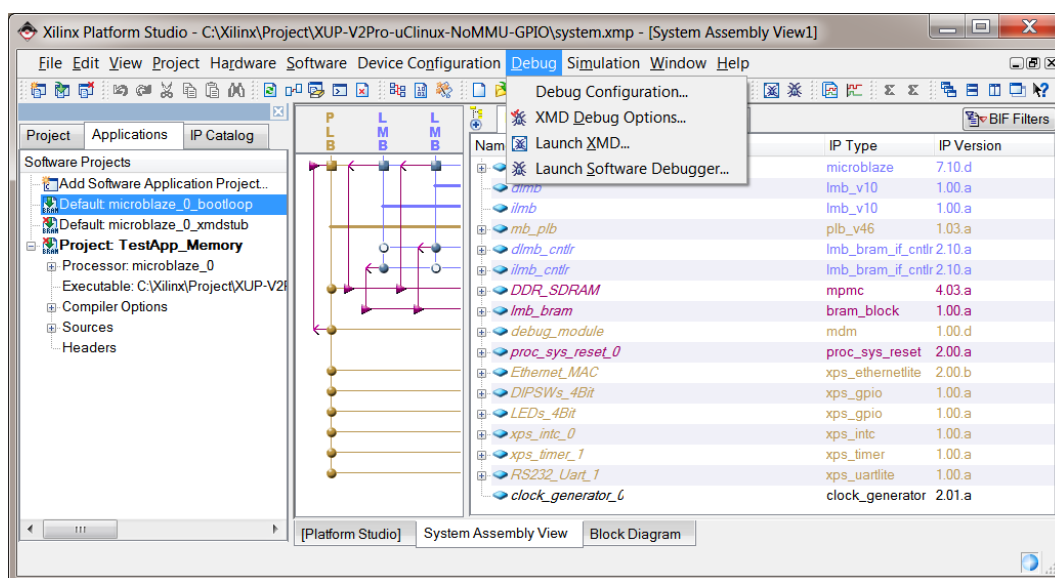
C.4 Postopek namestitve slike operacijskega in datotečnega sistema ter poskusni zagon sistema

```
1 [root@devpltfm petalinux-dist]# cp images/image.bin /mnt/
```

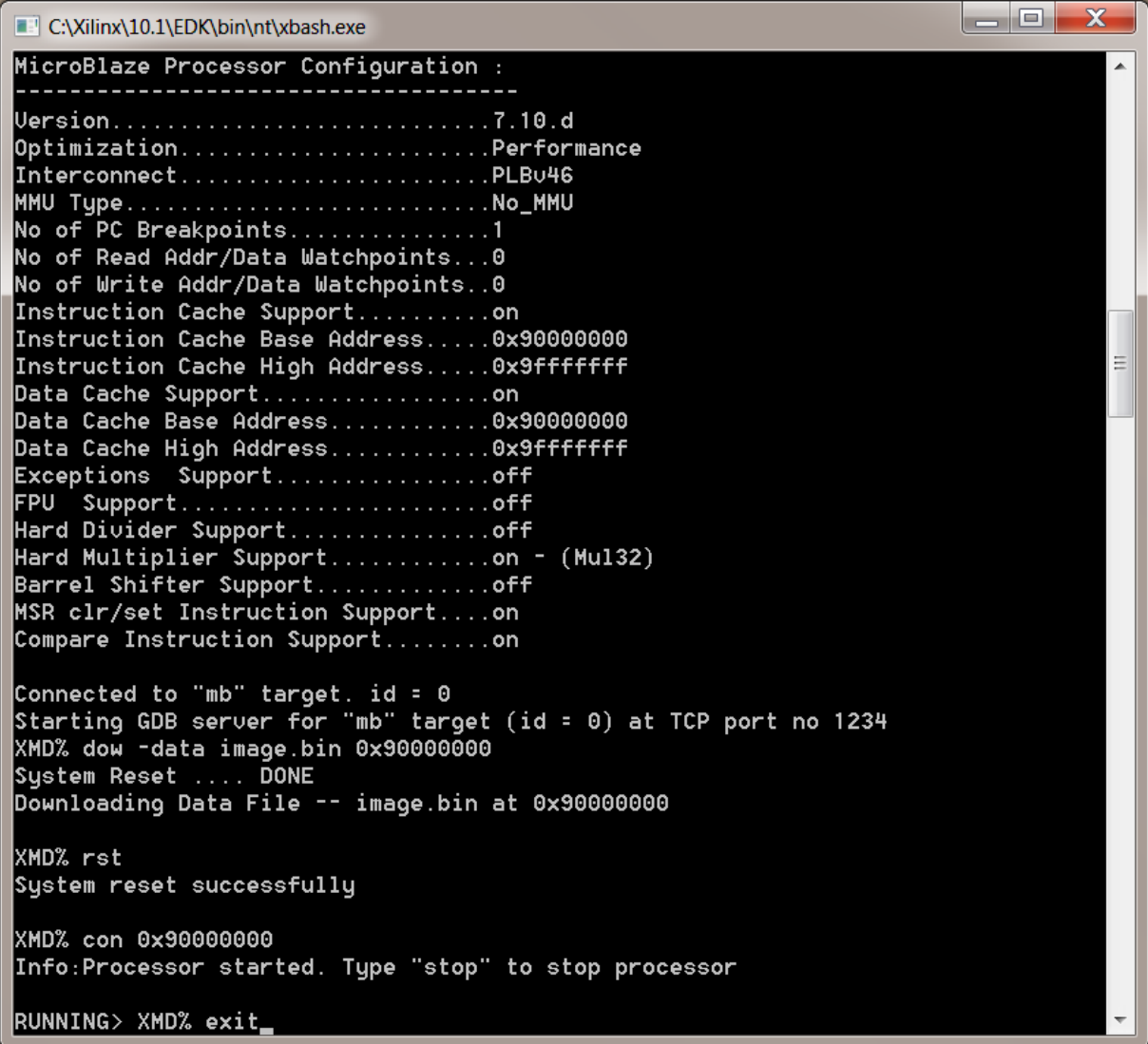
Izpis C.7: UKAZNA LUPINA - Prekopiranje slike v mapo projekta EDK



Slika C.15: Namestitev bitne slike z izbrano aplikacijo *bootloop* na sistem FPGA z uporabo povezave USB/JTAG (*Download bitstream*).



Slika C.16: Zagon razhroševalne konzole (*Launch XMD*) in samodejna povezava na razhroševalno enoto procesorja MicroBlaze.



```
C:\Xilinx\10.1\EDK\bin\nt\xbash.exe
MicroBlaze Processor Configuration :
-----
Version.....7.10.d
Optimization.....Performance
Interconnect.....PLBu46
MMU Type.....No_MMU
No of PC Breakpoints.....1
No of Read Addr/Data Watchpoints...0
No of Write Addr/Data Watchpoints..0
Instruction Cache Support.....on
Instruction Cache Base Address.....0x90000000
Instruction Cache High Address.....0x9fffffff
Data Cache Support.....on
Data Cache Base Address.....0x90000000
Data Cache High Address.....0x9fffffff
Exceptions Support.....off
FPU Support.....off
Hard Divider Support.....off
Hard Multiplier Support.....on - (Mul32)
Barrel Shifter Support.....off
MSR clr/set Instruction Support....on
Compare Instruction Support.....on

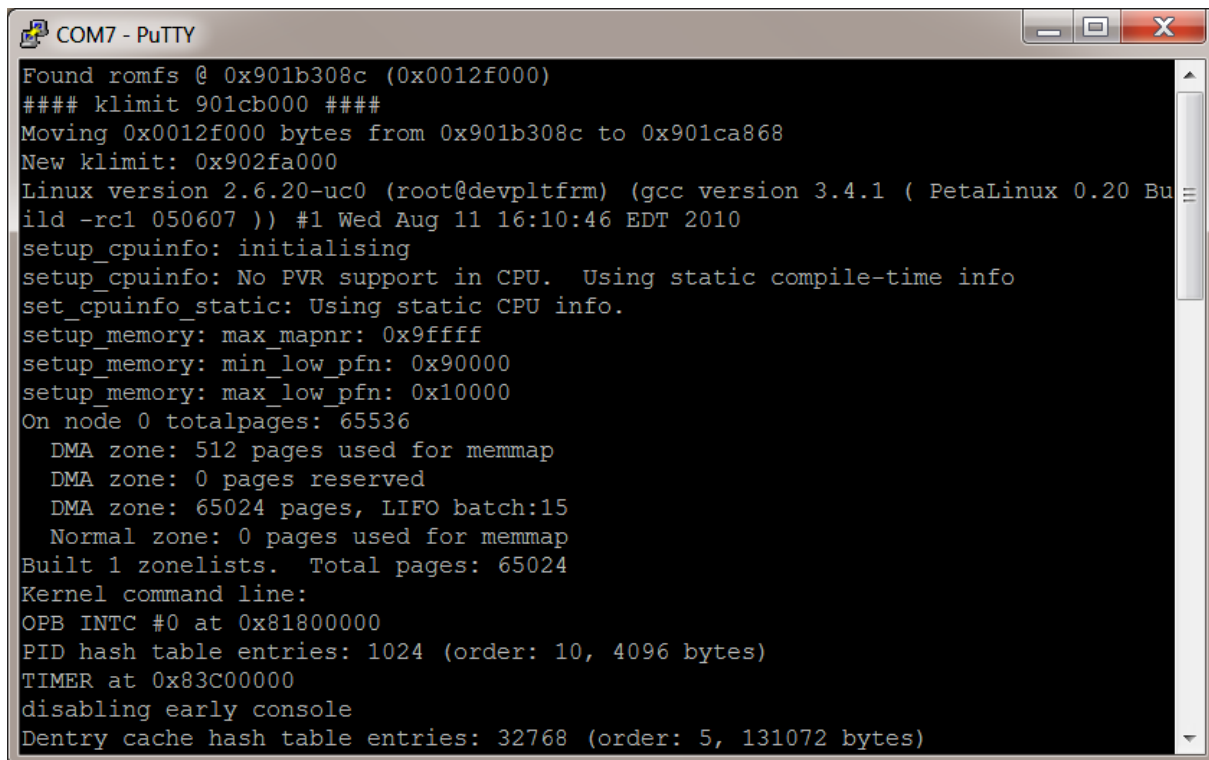
Connected to "mb" target. id = 0
Starting GDB server for "mb" target (id = 0) at TCP port no 1234
XMD% dow -data image.bin 0x90000000
System Reset .... DONE
Downloading Data File -- image.bin at 0x90000000

XMD% rst
System reset successfully

XMD% con 0x90000000
Info:Processor started. Type "stop" to stop processor

RUNNING> XMD% exit_
```

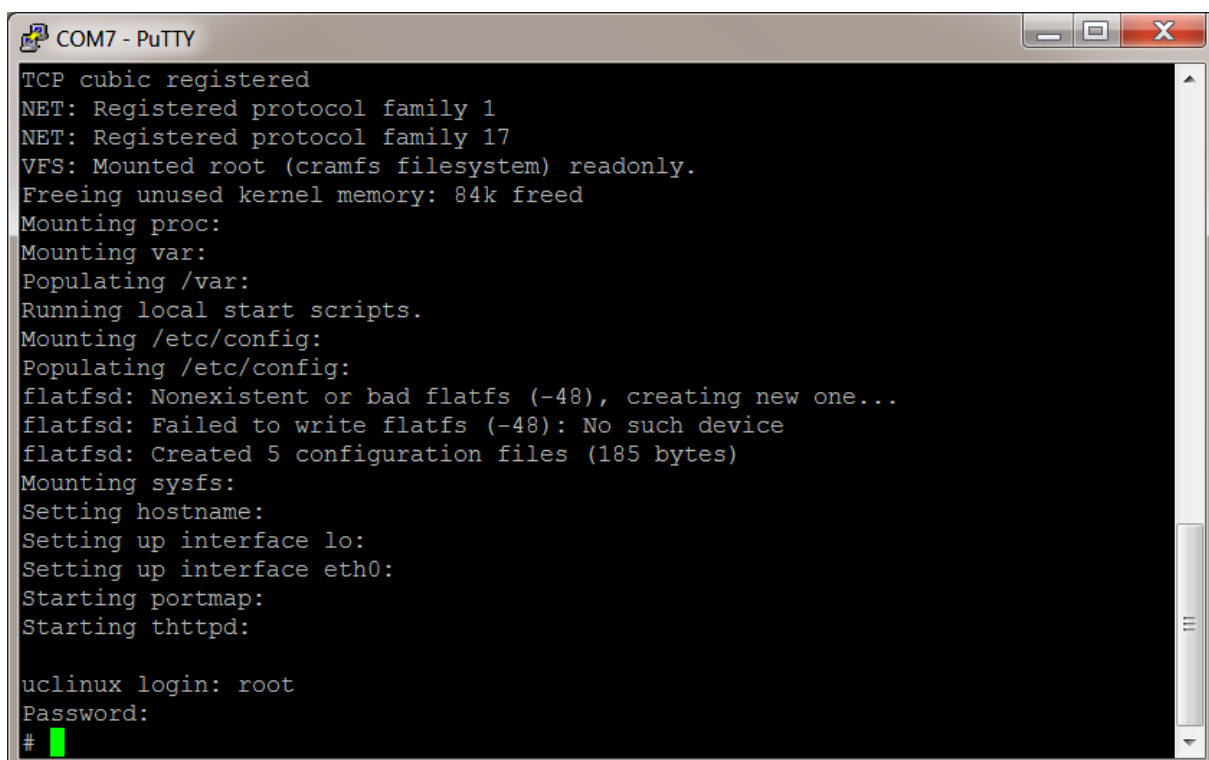
Slika C.17: Nalaganje slike operacijskega in datotečnega sistema `image.bin` v pomnilnik RAM ciljnega sistema z začetnim naslovom `0x90000000` (*proces nalaganja je dolgotrajen, privoščite si kavo*), ponastavitev procesorja in nastavitve ukaznega kazalca na naslov `0x90000000`.



```

COM7 - PuTTY
Found romfs @ 0x901b308c (0x0012f000)
#### klinit 901cb000 ####
Moving 0x0012f000 bytes from 0x901b308c to 0x901ca868
New klinit: 0x902fa000
Linux version 2.6.20-uc0 (root@devpltfm) (gcc version 3.4.1 ( PetaLinux 0.20 Bu
ild -rc1 050607 )) #1 Wed Aug 11 16:10:46 EDT 2010
setup_cpuinfo: initialising
setup_cpuinfo: No PVR support in CPU. Using static compile-time info
set_cpuinfo_static: Using static CPU info.
setup_memory: max_mapnr: 0x9ffff
setup_memory: min_low_pfn: 0x90000
setup_memory: max_low_pfn: 0x10000
On node 0 totalpages: 65536
  DMA zone: 512 pages used for memmap
  DMA zone: 0 pages reserved
  DMA zone: 65024 pages, LIFO batch:15
  Normal zone: 0 pages used for memmap
Built 1 zonelists. Total pages: 65024
Kernel command line:
OPB INTC #0 at 0x81800000
PID hash table entries: 1024 (order: 10, 4096 bytes)
TIMER at 0x83C00000
disabling early console
Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)
  
```

Slika C.18: Z malo sreče se izpiše zagonska sekvenca operacijskega sistema uClinux.



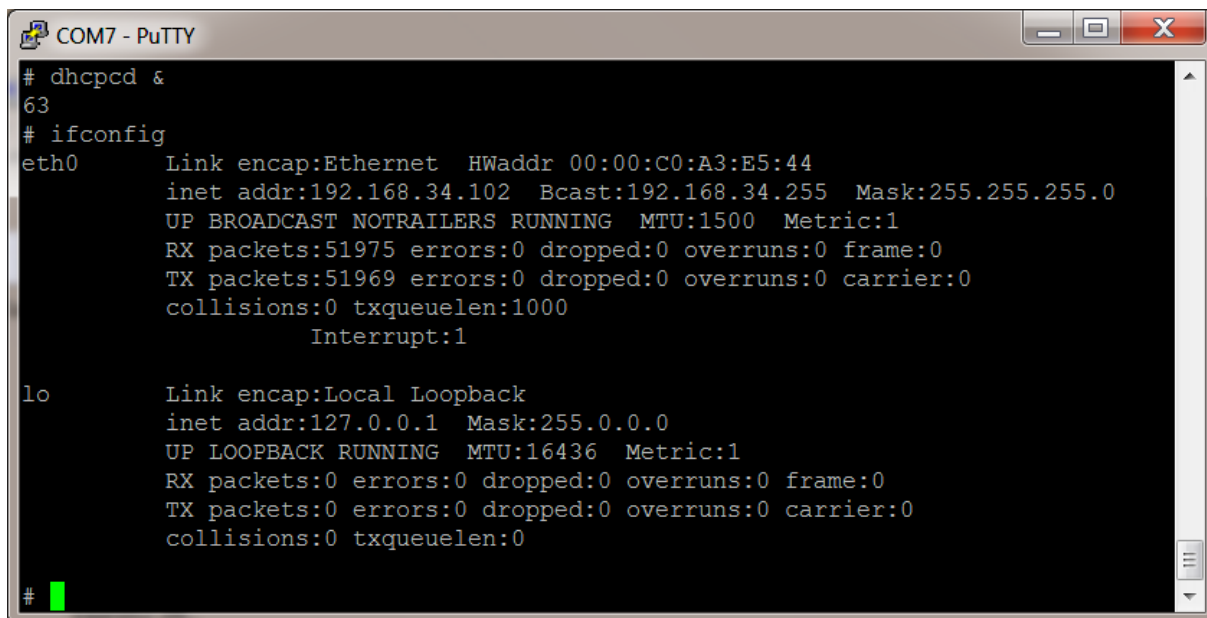
```

COM7 - PuTTY
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
VFS: Mounted root (cramfs filesystem) readonly.
Freeing unused kernel memory: 84k freed
Mounting proc:
Mounting var:
Populating /var:
Running local start scripts.
Mounting /etc/config:
Populating /etc/config:
flatfsd: Nonexistent or bad flatfs (-48), creating new one...
flatfsd: Failed to write flatfs (-48): No such device
flatfsd: Created 5 configuration files (185 bytes)
Mounting sysfs:
Setting hostname:
Setting up interface lo:
Setting up interface eth0:
Starting portmap:
Starting tthttpd:

uclinux login: root
Password:
#
  
```

Slika C.19: Zagonska sekvenca se konča s prijavo v sistem. Privzeto geslo uporabnika *root* je *root*.

C.5 Preizkus omrežne povezljivosti

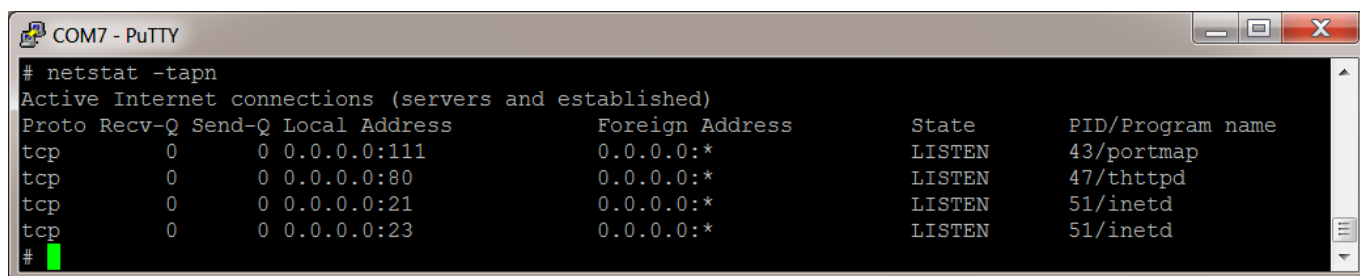


```
COM7 - PuTTY
# dhcpcd &
63
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:00:C0:A3:E5:44
          inet addr:192.168.34.102  Bcast:192.168.34.255  Mask:255.255.255.0
          UP BROADCAST NOTRAILERS RUNNING  MTU:1500  Metric:1
          RX packets:51975 errors:0 dropped:0 overruns:0 frame:0
          TX packets:51969 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
              Interrupt:1

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0

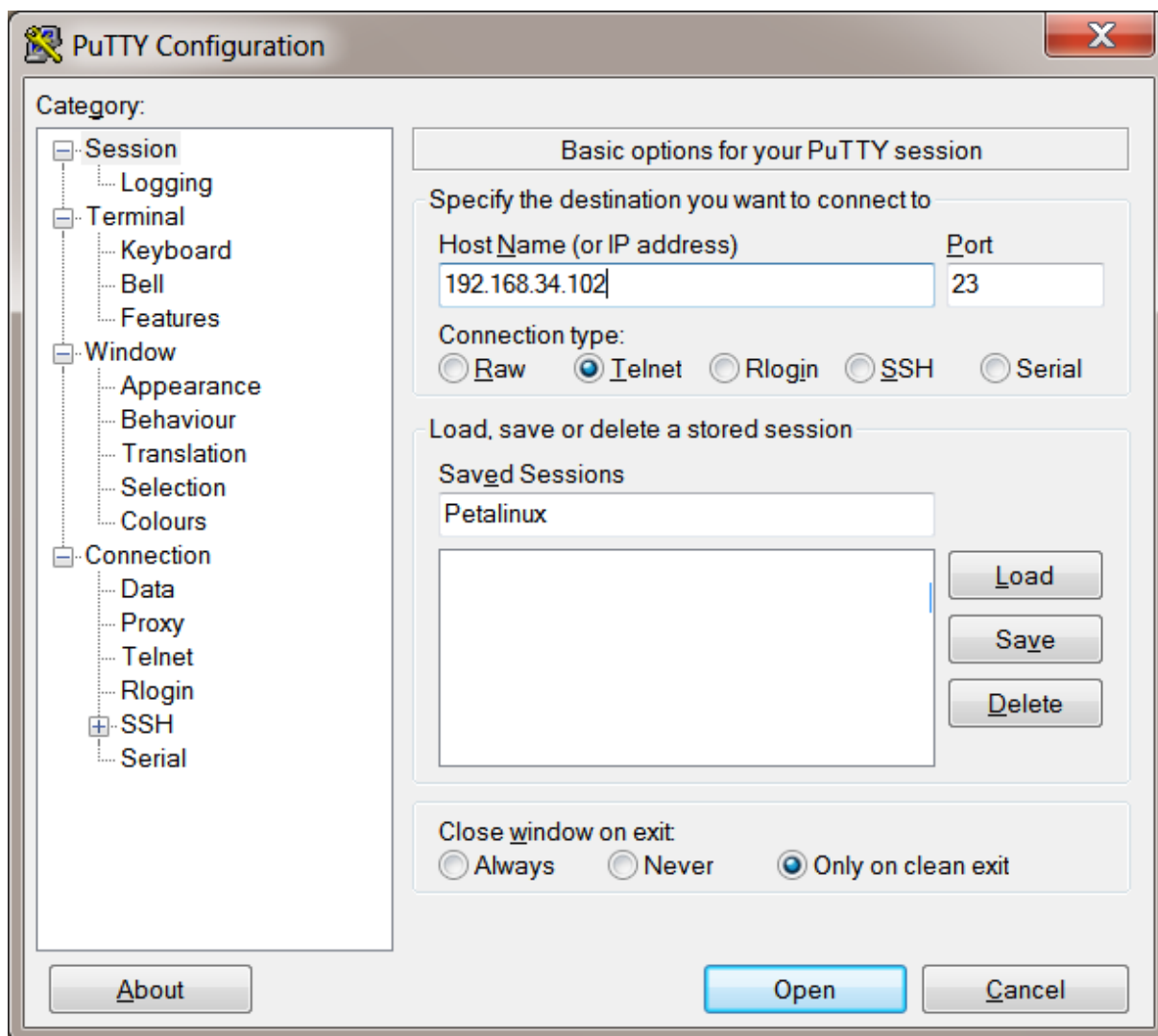
#
```

Slika C.20: V primeru, da je ciljni sistem v omrežju s storitvijo *DHCP* se nastavitve mrežne naprave izvede z uporabo programa *dhcpcd*. Uporaba znaka *&* pošlje proces v ozadje. Pridobljeni naslov IP se izpiše z uporabo ukaza *ifconfig*. V primeru, da storitev *DHCP* ni na voljo, se mrežno napravo nastavi z uporabo ukaza *ifconfig*.

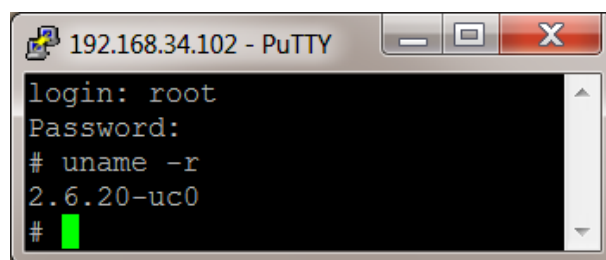


```
# netstat -tapn
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      43/portmap
tcp        0      0 0.0.0.0:80             0.0.0.0:*               LISTEN      47/thttpd
tcp        0      0 0.0.0.0:21             0.0.0.0:*               LISTEN      51/inetd
tcp        0      0 0.0.0.0:23             0.0.0.0:*               LISTEN      51/inetd
#
```

Slika C.21: uClinux ima privzeto zagnane naslednje mrežne storitve: Portmap, HTTP, FTP in Telnet.

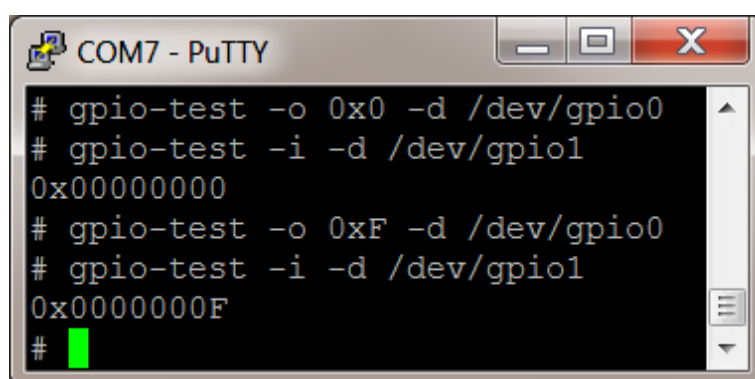


Slika C.22: Z uporabo odjemalca Telnet (program PuTTY) je mogoč oddaljen dostop do ciljnega sistema, če se le oba nahajata znotraj iste omrežne skupine.

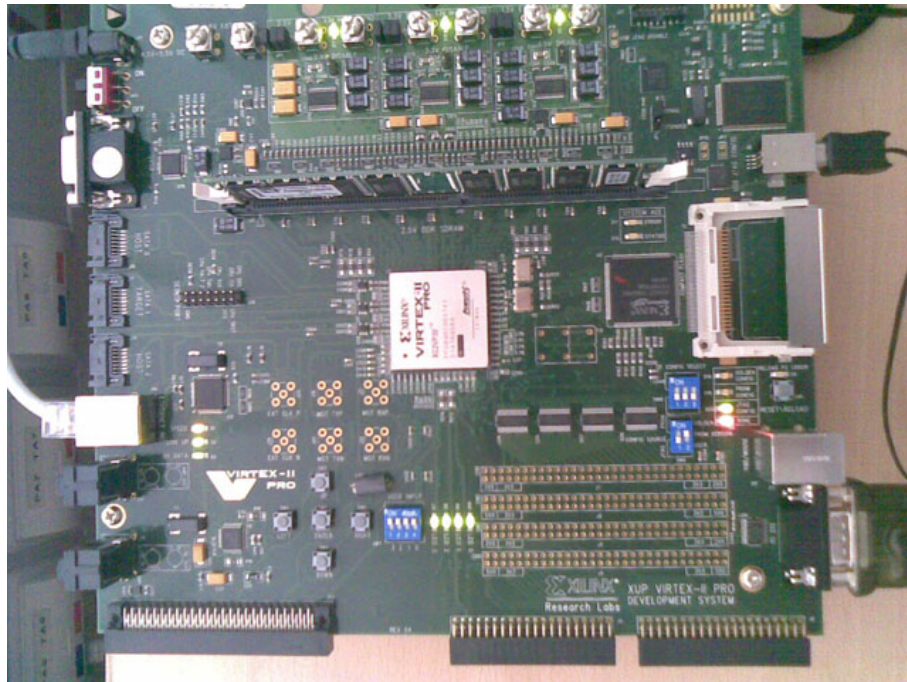


Slika C.23: Z malo sreče ☺ se oddaljena prijava v ciljni sistem prav tako uspešno zaključi.

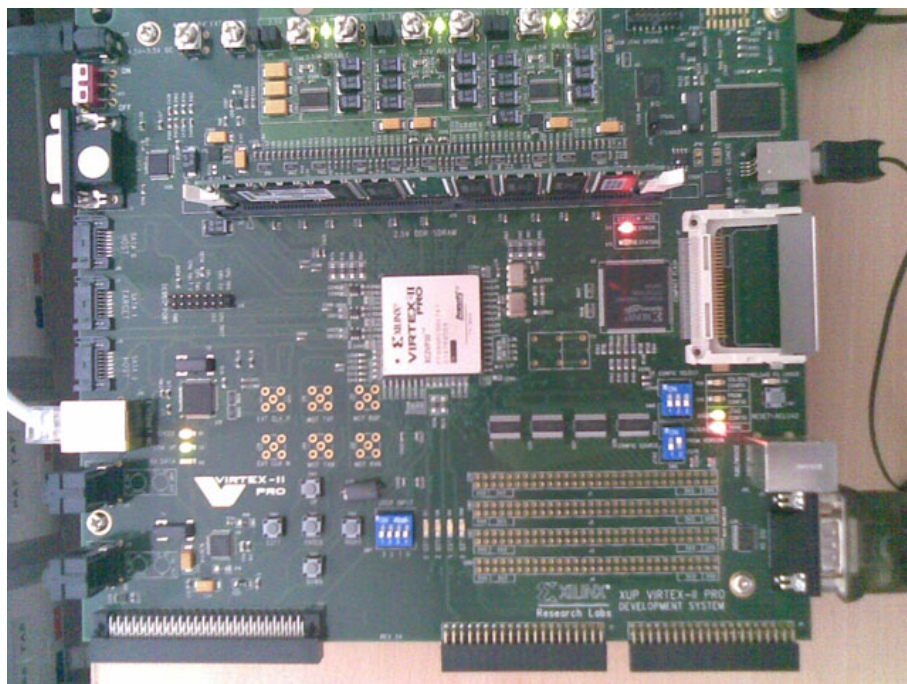
C.6 Preizkus naprave GPIO



Slika C.24: Preizkus naprav GPIO s programom *gpio-test*. Posledica prvih dveh ukazov so prižgane diode LED in detekcija nizkega logičnega stanja stikala C.25, posledica drugih dveh pa ugasnjene diode LED in detekcija visokega logičnega stanja stikala C.26.



Slika C.25: Test razvojnega sistema FPGA (1).



Slika C.26: Test razvojnega sistema FPGA (2).

C.7 Povzetek poglavja

Vsebina diagrama A.5, sinhronizacija strojne opreme z izvorno kodo operacijskega sistema in definicija ter gradnja operacijskega sistema, je bila podrobno predstavljen v odseku C.1, C.2 in C.3.

Vsebina diagrama A.6, postopek namestitve operacijskega sistema na strojno opremo, je bila podrobno predstavljena v odseku C.4.

Vsebina diagrama A.7, preizkus dostopa preko serijskega vmesnika in oddaljen dostop do ciljnega sistema, je bila podrobno predstavljena v odseku C.4 in C.5.

V tem poglavju sta še prav tako bili preizkušeni namenski strojni napravi GPIO.

Izvorna koda nizkonivojskega gonilnika GPIO je bila prevzeta z projekta gradnje programske opreme EDK (Xilinx) in se skupaj z izvorno kodo visokonivojskega gonilnika (Petalinux) nahaja v naslednji mapi:

```
$PETALINUX/software/linux-2.6.x-petalogix/drivers/char/xilinx_gpio.
```

Vsi opravljeni postopki in testi predstavljajo potreben pogoj za namestitev in uporabo posebne aplikacije, ki bo predstavljena v naslednjem poglavju.

Prilagajanje postopka gradnje operacijskega sistema in dodajanje novih modulov jedru operacijskega sistema so opisani na strani podjetja PetaLinux.



Postopek načrtovanja, razvoja in namestitve testne aplikacije v datotečni sistem PetaLinux

D.1 Opis testne aplikacije

Odjemalec (CentOS 5.3) bo klical oddaljeno proceduro na ciljnem sistemu (uClinux 2.6). Argument funkcije oddaljene procedure je arbitrarni niz, ki ga nato ciljni sistem zapiše v datoteko `newfile`.

D.2 Gradnja in namestitvev teste aplikacije

Gradnja testne aplikacije za ciljni sistem poteka v razvojnem okolju za programsko opremo (CentOS 5.3). Namestitev slike operacijskega in datotečnega sistema na razvojni sistem FPGA poteka v razvojnem okolju za strojno opremo (Windows 7).

```
1 [root@devpltfm petalinux-dist]# petalinux-new-app EmbeddedTestApp1
2 New application template successfully created in software/user-apps/EmbeddedTestApp1
3 See EmbeddedTestApp1/README for what to do next.
4 [root@devpltfm petalinux-dist]# cd ../../software/user-apps/EmbeddedTestApp1/
5 [root@devpltfm EmbeddedTestApp1]# ls
6 Makefile  README  EmbeddedTestApp1.c  msg_clnt.c  msg.h
```

Izpis D.1: UKAZNA LUPINA - Kreacija projekta testne aplikacije.

```
1 /*
2  * EmbeddedTestApp1
3  */
4
5 #include <stdio.h>
6 #include <rpc/rpc.h> /* always needed */
7 #include "msg.h"     /* msg.h will be generated by rpcgen */
8
9 /*
10  * EmbeddedTestApp1
11  */
12 int *
```

```

13 printmessage_1_svc(char **msg, struct svc_req* cl)
14 {
15     static int result; /* must be static! */
16     FILE *f;
17
18     f = fopen("newfile", "w");
19     if (f == NULL) {
20         result = 0;
21         return (&result);
22     }
23
24     fprintf(f, "%s\n", *msg);
25     fclose(f);
26     result = 1;
27     return (&result);
28 }

```

Izpis D.2: Vsebina datoteke EmbeddedTestApp1.c.

```

1  /*
2   * Please do not edit this file.
3   * It was generated using rpcgen.
4   */
5
6  #include "msg.h"
7  #include <stdio.h>
8  #include <stdlib.h>
9  #include <rpc/pmap_clnt.h>
10 #include <string.h>
11 #include <memory.h>
12 #include <sys/socket.h>
13 #include <netinet/in.h>
14
15 #ifndef SIG_PF
16 #define SIG_PF void(*) (int)
17 #endif
18
19 static void
20 messageprog_1(struct svc_req *rqstp, register SVCXPRT *transp)
21 {
22     union {
23         char *printmessage_1_arg;
24     } argument;
25     char *result;
26     xdrproc_t _xdr_argument, _xdr_result;
27     char *(*local)(char *, struct svc_req *);
28
29     switch (rqstp->rq_proc) {
30     case NULLPROC:
31         (void) svc_sendreply (transp, (xdrproc_t) xdr_void, (char *)NULL);
32         return;
33
34     case PRINTMESSAGE:
35         _xdr_argument = (xdrproc_t) xdr_wrapstring;
36         _xdr_result = (xdrproc_t) xdr_int;
37         local = (char *(*)(char *, struct svc_req *)) printmessage_1_svc;
38         break;
39
40     default:
41         svcerr_noproc (transp);
42         return;
43     }
44     memset ((char *)&argument, 0, sizeof (argument));
45     if (!svc_getargs (transp, (xdrproc_t) _xdr_argument, (caddr_t) &argument)) {
46         svcerr_decode (transp);
47         return;
48     }
49     result = (*local)((char *)&argument, rqstp);
50     if (result != NULL && !svc_sendreply(transp, (xdrproc_t) _xdr_result, result)) {

```

```

51     svcerr_systemerr (transp);
52 }
53 if (!svc_freeargs (transp, (xdrproc_t) _xdr_argument, (caddr_t) &argument)) {
54     fprintf (stderr, "%s", "unable_to_free_arguments");
55     exit (1);
56 }
57 return;
58 }
59
60 int
61 main (int argc, char **argv)
62 {
63     register SVCXPRT *transp;
64
65     pmap_unset (MESSAGEPROG, MESSAGEVERS);
66
67     transp = svcudp_create(RPC_ANYSOCK);
68     if (transp == NULL) {
69         fprintf (stderr, "%s", "cannot_create_udp_service.");
70         exit(1);
71     }
72     if (!svc_register(transp, MESSAGEPROG, MESSAGEVERS, messageprog_1, IPPROTO_UDP)) {
73         fprintf (stderr, "%s", "unable_to_register_(MESSAGEPROG,_MESSAGEVERS,_udp).");
74         exit(1);
75     }
76
77     transp = svctcp_create(RPC_ANYSOCK, 0, 0);
78     if (transp == NULL) {
79         fprintf (stderr, "%s", "cannot_create_tcp_service.");
80         exit(1);
81     }
82     if (!svc_register(transp, MESSAGEPROG, MESSAGEVERS, messageprog_1, IPPROTO_TCP)) {
83         fprintf (stderr, "%s", "unable_to_register_(MESSAGEPROG,_MESSAGEVERS,_tcp).");
84         exit(1);
85     }
86
87     svc_run ();
88     fprintf (stderr, "%s", "svc_run_returned");
89     exit (1);
90     /* NOTREACHED */
91 }

```

Izpis D.3: Vsebina datoteke msg_svc.c.

```

1  /*
2   * Please do not edit this file.
3   * It was generated using rpcgen.
4   */
5
6  #ifndef _MSG_H_RPCGEN
7  #define _MSG_H_RPCGEN
8
9  #include <rpc/rpc.h>
10
11  #ifdef __cplusplus
12  extern "C" {
13  #endif
14
15  #define MESSAGEPROG 0x20000099
16  #define MESSAGEVERS 1
17
18  #if defined(__STDC__) || defined(__cplusplus)
19  #define PRINTMESSAGE 1
20  extern int * printmessage_1(char **, CLIENT *);
21  extern int * printmessage_1_svc(char **, struct svc_req *);
22  extern int messageprog_1_freeresult (SVCXPRT *, xdrproc_t, caddr_t);
23
24  #else /* K&R C */
25  #define PRINTMESSAGE 1

```

```

26 extern int * printmessage_1();
27 extern int * printmessage_1_svc();
28 extern int messageprog_1_freeresult ();
29 #endif /* K&R C */
30
31 #ifdef __cplusplus
32 }
33 #endif
34
35 #endif /* !_MSG_H_RPCGEN */

```

Izpis D.4: Vsebina datoteke msg.h.

```

1 ifndef PETALINUX
2 $(error You must source the petalinux/settings.sh script before working with PetaLinux)
3 endif
4
5 # Point to default PetaLinux root directory
6 ifndef ROOTDIR
7 ROOTDIR=$(PETALINUX)/software/petalinux-dist
8 endif
9
10 PATH:=$(PATH):$(ROOTDIR)/tools
11
12 UCLINUX_BUILD_USER = 1
13 -include $(ROOTDIR)/.config
14 -include $(ROOTDIR)/$(CONFIG_LINUXDIR)/.config
15 LIBCDIR = $(CONFIG_LIBCDIR)
16 -include $(ROOTDIR)/config.arch
17 ROMFSDIR=$(ROOTDIR)/romfs
18 ROMFSINST=$(ROOTDIR)/tools/romfs-inst.sh
19
20 APP = EmbeddedTestAppl
21
22 # Add any other object files to this list below
23 APP_OBJS = EmbeddedTestAppl.o msg_svc.o
24
25 all: $(APP)
26
27 $(APP): $(APP_OBJS)
28 $(CC) $(LDFLAGS) -o $@ $(APP_OBJS) $(LDLIBS)
29
30 clean:
31 -rm -f $(APP) *.elf *.gdb *.o
32
33 .PHONY: romfs image
34
35 romfs:
36 $(ROMFSINST) $(APP) /bin/$(APP)
37
38 image: romfs
39 make -C ${PETALINUX}/software/petalinux-dist image
40
41 %.o: %.c
42 $(CC) -c $(CFLAGS) -o $@ $<
43
44
45 # Targets for the required .config files - if they don't exist, the tree isn't
46 # configured. Tell the user this, how to fix it, and exit.
47 ${ROOTDIR}/config.arch ${ROOTDIR}/.config:
48 @echo "Error:_You_must_configure_the_PetaLinux_tree_before_compiling_your_application"
49 @echo ""
50 @echo "Change_directory_to_../petalinux-dist_and_'make_menuconfig'_or_'make_xconfig'"
51 @echo ""
52 @echo "Once_the_tree_is_configured,_return_to_this_directory,_and_re-run_make."
53 @echo ""
54 @exit -1

```

Izpis D.5: Vsebina datoteke Makefile.

```
1 [root@devpltfm EmbeddedTestApp]# make && make romfs
2 /root/uclinux/petalinux-v0.40-final/software/petalinux-dist/tools/romfs-inst.sh →
   ↪ EmbeddedTestApp /bin/EmbeddedTestApp
```

Izpis D.6: UKAZNA LUPINA - Gradnja testne aplikacije in namestitvev v datotečni sistem.

Opomba: Datoteka `/etc/protocols` v različici `petalinux-0.40` manjka, zato se jo prekopira z razvojnega sistema CentOS 5.3. V nasprotnem primeru storitev RPC ne bo delovala.

```
1 [root@devpltfm petalinux-dist]# cp /etc/protocols →
   ↪ $PETALINUX/software/petalinux-dist/romfs/etc
```

Izpis D.7: UKAZNA LUPINA - Kopiranje datoteke `protocols` v datotečni sistem ciljnega sistema.

```
1 [root@devpltfm EmbeddedTestApp]# cd $PETALINUX/software/petalinux-dist/
2 [root@devpltfm petalinux-dist]# make romfs
3 [root@devpltfm petalinux-dist]# make image
```

Izpis D.8: UKAZNA LUPINA - Gradnja uporabniške programske opreme in posodobitev binarne slike operacijskega in datotečnega sistema.

Slika operacijskega in datotečnega sistema se naloži na razvojni sistem (ponovi postopke odseka C.4) in nastavi se mrežno napravo (ponovi postopke odseka C.5).

Manjka slika zagnanega ciljnega sistema, nastavljene mrežne naprave in delujočega strežniškega procesa RPC.

D.3 Gradnja odjemalca za testno aplikacijo

Gradnja odjemalca za testno aplikacijo poteka v razvojnem okolju za programsko opremo (CentOS 5.3). Aplikacija odjemalca se ne bo prevajala z pridobljenimi orodji EDK (križni prevajalnik), ampak z nativnimi orodji, saj je aplikacija namenjena sistemu CentOS 5.3.

```
1 [root@devpltfm ~]# mkdir EmbeddedTestApp1Client
2 [root@devpltfm ~]# cd EmbeddedTestApp1Client/
3 [root@devpltfm EmbeddedTestApp1Client]# ls
4 EmbeddedTestApp1Client.c msg_clnt.c msg.h
```

Izpis D.9: UKAZNA LUPINA - Kreacija projekta odjemalca za testno aplikacijo.

```
1 /*
2  * Embedded Test Application
3  */
4
5 #include <stdio.h>
6 #include <rpc/rpc.h>      /* always needed */
7 #include "msg.h"         /* msg.h will be generated by rpcgen */
8
9 void
10 main(int argc, char* argv[])
11 {
12     CLIENT *cl;
13     int *result;
14     char *server;
15     char *message;
16
17     if (argc != 3) {
18         fprintf(stderr, "usage:_%s_host_message\n", argv[0]);
19         exit(1);
20     }
21
22     server = argv[1];
23     message = argv[2];
24
25     /*
26      * Create client "handle" used for calling MESSAGEPROG on
27      * the server designated on the command line. We tell
28      * the RPC package to use the TCP protocol when
29      * contacting the server.
30      */
31
32     cl = clnt_create(server, MESSAGEPROG, MESSAGEVERS, "tcp");
33     if (cl == NULL) {
34         /*
35          * Couldn't establish connection with server.
36          * Print error message and stop.
37          */
38
39         clnt_pcreateerror(server);
40         exit(1);
41     }
42
43     /*
44      * Call the remote procedure "printmessage" on the server
45      */
46     result = printmessage_1(&message, cl);
47     if (result == NULL) {
48         /*
49          * An error occurred while calling the server.
50          * Print error message and stop.
51          */
52
```

```

53         clnt_perror(cl, server);
54         exit(1);
55     }
56
57     /*
58     * Okay, we successfully called the remote procedure.
59     */
60
61     if (*result == 0) {
62
63         /*
64         * Server was unable to print our message.
65         * Print error message and stop.
66         */
67         fprintf(stderr, "%s:_%s_couldn't_print_your_message\n",
68             argv[0], server);
69         exit(1);
70     }
71
72     /*
73     * The message got printed on the server
74     */
75
76     printf("Message_delivered_to_%s!\n", server);
77     exit(0);
78 }

```

Izpis D.10: Vsebina datoteke EmbeddedTestApp1Client.c.

```

1  /*
2  * Please do not edit this file.
3  * It was generated using rpcgen.
4  */
5
6  #include <memory.h> /* for memset */
7  #include "msg.h"
8
9  /* Default timeout can be changed using clnt_control() */
10 static struct timeval TIMEOUT = { 25, 0 };
11
12 int *
13 printmessage_1(char **argp, CLIENT *clnt)
14 {
15     static int clnt_res;
16
17     memset((char *)&clnt_res, 0, sizeof(clnt_res));
18     if (clnt_call(clnt, PRINTMESSAGE,
19         (xdrproc_t) xdr_wrapstring, (caddr_t) argp,
20         (xdrproc_t) xdr_int, (caddr_t) &clnt_res,
21         TIMEOUT) != RPC_SUCCESS) {
22         return (NULL);
23     }
24     return (&clnt_res);
25 }

```

Izpis D.11: Vsebina datoteke msg_clnt.c.

```

1  /*
2  * Please do not edit this file.
3  * It was generated using rpcgen.
4  */
5
6  #ifndef _MSG_H_RPCGEN
7  #define _MSG_H_RPCGEN
8
9  #include <rpc/rpc.h>
10
11 #ifdef __cplusplus
12 extern "C" {

```

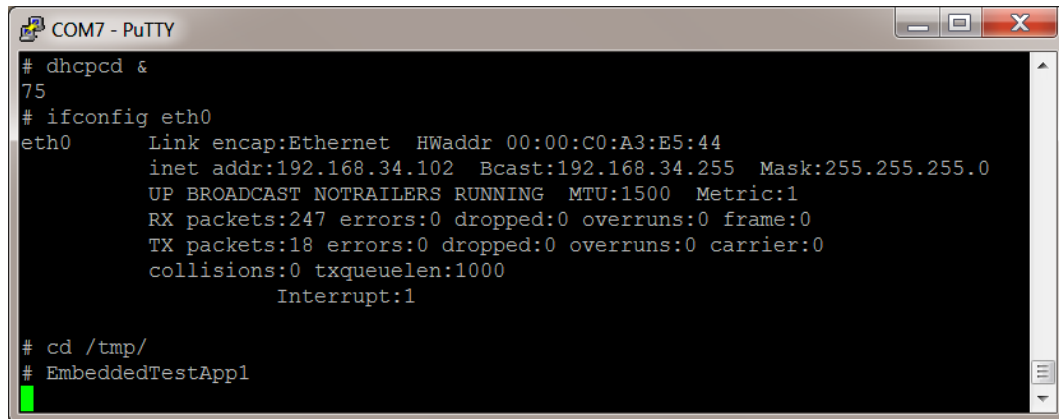
```
13 #endif
14
15 #define MESSAGEPROG 0x20000099
16 #define MESSAGEVERS 1
17
18 #if defined(__STDC__) || defined(__cplusplus)
19 #define PRINTMESSAGE 1
20 extern int * printmessage_1(char **, CLIENT *);
21 extern int * printmessage_1_svc(char **, struct svc_req *);
22 extern int messageprog_1_freeresult (SVCXPRT *, xdrproc_t, caddr_t);
23
24 #else /* K&R C */
25 #define PRINTMESSAGE 1
26 extern int * printmessage_1();
27 extern int * printmessage_1_svc();
28 extern int messageprog_1_freeresult ();
29 #endif /* K&R C */
30
31 #ifdef __cplusplus
32 }
33 #endif
34
35 #endif /* !_MSG_H_RPCGEN */
```

Izpis D.12: Vsebina datoteke msg.h.

```
1 [root@devpltfm EmbeddedTestApplClient]# gcc EmbeddedTestApplClient.c msg_clnt.c -o →
   ↪ EmbeddedTestApplClient
```

Izpis D.13: UKAZNA LUPINA - Gradnja odjemalca za testno aplikacijo.

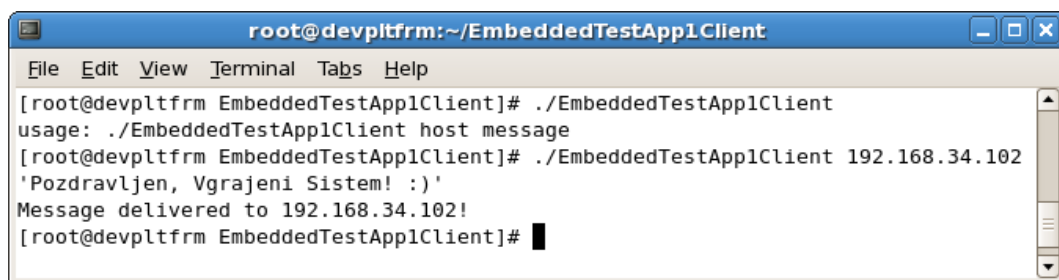
D.4 Preizkus testne aplikacije



```
# dhcpd &
75
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:00:C0:A3:E5:44
          inet addr:192.168.34.102  Bcast:192.168.34.255  Mask:255.255.255.0
          UP BROADCAST NOTRAILERS RUNNING MTU:1500  Metric:1
          RX packets:247 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          Interrupt:1

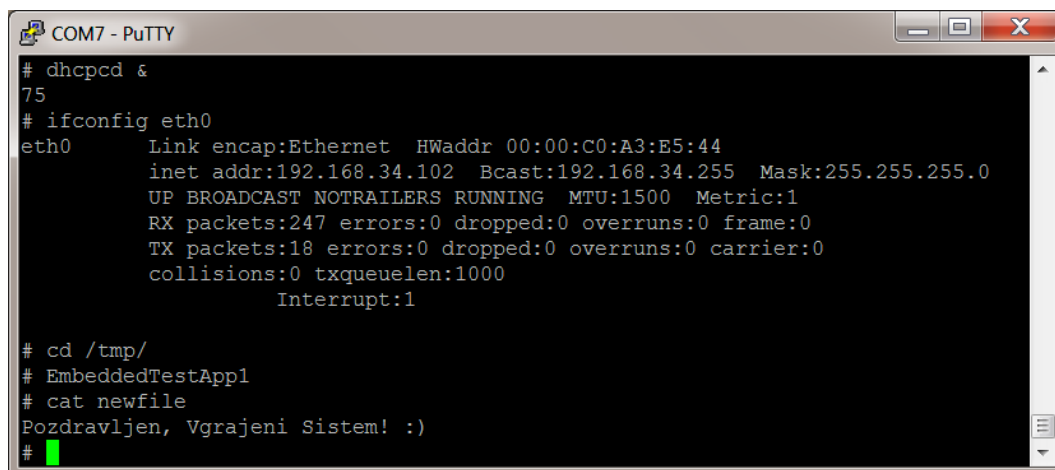
# cd /tmp/
# EmbeddedTestAppl
```

Slika D.1: Nastavitev mrežne naprave in kreacija procesa strežnika RPC testne aplikacije na ciljnem sistemu.



```
root@devpltfm:~/EmbeddedTestApp1Client
File Edit View Terminal Tabs Help
[root@devpltfm EmbeddedTestApp1Client]# ./EmbeddedTestApp1Client
usage: ./EmbeddedTestApp1Client host message
[root@devpltfm EmbeddedTestApp1Client]# ./EmbeddedTestApp1Client 192.168.34.102
'Pozdravljen, Vgrajeni Sistem! :)'
Message delivered to 192.168.34.102!
[root@devpltfm EmbeddedTestApp1Client]#
```

Slika D.2: Klic oddaljene procedure s sistema CentOS.



```
# dhcpcd &
75
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:00:C0:A3:E5:44
          inet addr:192.168.34.102  Bcast:192.168.34.255  Mask:255.255.255.0
          UP BROADCAST NOTRAILERS RUNNING MTU:1500  Metric:1
          RX packets:247 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
              Interrupt:1

# cd /tmp/
# EmbeddedTestAppl
# cat newfile
Pozdravljen, Vgrajeni Sistem! :)
#
```

Slika D.3: Preverjanje pravilnosti izvedbe programa RPC testne aplikacije na ciljnem sistemu.

D.5 Povzetek poglavja

Vsebina diagrama A.8, gradnja in namestitev testne aplikacije za ciljni sistem je bila podrobno predstavljena v odseku D.2. V odseku D.3 je bila podrobno predstavljena gradnja in namestitev odjemalca za testno aplikacijo za CentOS 5.3. Postopek gradnje testne aplikacije je potekal v razvojnem okolju za programsko opremo (CentOS) pri tem pa so glede na namen uporabljala različna orodja za gradnjo programov.

E.1 Primer opisa značke

```

1  -----
2  -- Company: UL FRI
3  -- Engineer: Iztok Starc
4  --
5  -- Revision:
6  -- Revision 0.01 - File Created
7  -- Additional Comments:
8  --
9  -----
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.STD_LOGIC_ARITH.ALL;
13 use IEEE.STD_LOGIC_UNSIGNED.ALL;
14
15 entity rfid_transponder_emu is
16     Port ( clk : in  STD_LOGIC;
17           reset : in  STD_LOGIC;
18
19           emu_nse : in STD_LOGIC_VECTOR(27 downto 0);
20           emu_nse_select : in STD_LOGIC;
21           emu_ce : in STD_LOGIC;
22           emu_rfid_state : out STD_LOGIC_VECTOR(30 downto 0);
23
24           rx : in STD_LOGIC;
25           tx : out STD_LOGIC
26     );
27 end rfid_transponder_emu;
28
29 architecture Example of rfid_transponder_emu is
30
31     --! @brief RFID TRANSPONDER
32     --! @description Explicitly Encoded FSM States
33     constant STATE_LENGTH : integer := 3;
34     constant S0 : std_logic_vector(2 downto 0) := "000";
35     constant S1 : std_logic_vector(2 downto 0) := "001";
36     constant S2 : std_logic_vector(2 downto 0) := "010";
37     constant S3 : std_logic_vector(2 downto 0) := "011";
38
39     signal cs, ns, nsi, nse : std_logic_vector(2 downto 0);
40
41 begin
42
43     --!
44     --! @brief RFID Implementation Example & Integration with the Emulation Stub

```

```

45  --!
46
47  --!
48  --! @brief Next State Logic
49
50  --! @brief Next State Logic: Internal
51  NEXT_STATE_INTERNAL_LOGIC: process(cs, rx)
52  begin
53      case cs is
54          when S0 =>
55              if rx = '1' then
56                  nsi <= S1;
57              else
58                  nsi <= S0;
59              end if;
60
61          when S1 =>
62              nsi <= S2;
63
64          when S2 =>
65              nsi <= S3;
66
67          when others =>
68              nsi <= S0;
69
70      end case;
71  end process;
72
73  --! @brief Next State Logic: External (###EMU Stub###)
74  NEXT_STATE_EXTERNAL_LOGIC: nse <= emu_nse(STATE_LENGTH-1 downto 0); -- Next State External
75  NEXT_STATE_LOGIC: ns <= nse when emu_nse_select = '1' else -- If selected, Next State →
76      ↪ External,                                nsi;                                -- else, Next State Internal
77
78  --!
79  --! @brief Sequential Logic
80  TRANSPONDER_SEQ_LOGIC: process(clk)
81  begin
82      if clk'event and clk = '1' then
83          if reset = '1' then
84              cs <= S0;
85          elsif emu_ce = '1' then
86              cs <= ns;
87          end if;
88      end if;
89  end process;
90
91  --!
92  --! @brief Output Logic
93
94  --! @brief Output Logic (###EMU Stub###)
95  emu_rfid_state <= conv_std_logic_vector(0, emu_rfid_state'length-cs'length) & cs;
96
97  --! @brief Output Logic: Output
98  OUTPUT_LOGIC: process(cs)
99  begin
100      case cs is
101          when S0 =>
102              tx <= '0';
103
104          when S1 =>
105              tx <= '0';
106
107          when S2 =>
108              tx <= '1';
109
110          when others =>
111              tx <= '0';
112
113      end case;

```

```

114 end process;
115
116 end Example;

```

Izpis E.1: Primer opisa značke v VHDL.

E.2 Primer opisa protokolnega analizatorja

```

1 package sectool.bridgeControl;
2
3 public class SecToolMessageTransformation {
4
5     /**
6      * SecTool -> REE
7      */
8     public static final int ARGMASK_REE_RX      = 0x00000001;
9     public static final int ARGMASK_REE_FSM     = 0x0000000E;
10    public static final int ARGMASK_REE_OPCODE = 0xE0000000;
11
12    public static final int ARGFLAG_REE_OPCODE_NSi_TXi = 0x00000000;
13    public static final int ARGFLAG_REE_OPCODE_NSi_TXe = 0x20000000;
14    public static final int ARGFLAG_REE_OPCODE_NSe_TXi = 0x40000000;
15    public static final int ARGFLAG_REE_OPCODE_NSe_TXe = 0x60000000;
16    public static final int ARGFLAG_REE_OPCODE_RESET  = 0xE0000000;
17
18    /**
19     * SecTool -> Reader
20     */
21    public static final int RETMASK_REE_TX      = 0x00000001;
22    public static final int RETMASK_REE_FSM     = 0x0000000E;
23    public static final int RETMASK_REE_FSM_S0 = 0x00000000;
24    public static final int RETMASK_REE_FSM_S1 = 0x00000002;
25    public static final int RETMASK_REE_FSM_S2 = 0x00000004;
26    public static final int RETMASK_REE_FSM_S3 = 0x00000006;
27
28    /**
29     *
30     * @param messages
31     * @param padding
32     * @return
33     */
34    public static int[] compileReader2REEBridgeMessages(byte[] messages, int padding)
35        throws RuntimeException {
36
37        if(messages == null || messages.length > 1 || padding < 0)
38            throw new RuntimeException("Oops!_compileReader2REEBridgeMessages_passed →
39                ↳ _argument_verification_failed");
40
41        int[] result = new int[messages.length*8+padding];
42
43        for(int i=0; i<messages.length*8; i++) {
44
45            /* Extract bit */
46            int rxBitMask = 1; /* Create bit mask */
47            int tmpResult = 0; /* Temporary result */
48
49            rxBitMask <= i%8; /* Shift bit to appropriate position */
50            tmpResult = messages[0] & rxBitMask; /* Bitwise AND */
51            tmpResult >= i%8; /* Shift to appropriate position */
52
53            tmpResult |= SecToolMessageTransformation.ARGFLAG_REE_OPCODE_NSi_TXe; /* →
54                ↳ Add appropriate opcode */
55
56            result[i] = tmpResult;
57        }
58    }
59 }

```

```

56
57         for(int i=0; i<padding; i++) {
58             result[i+messages.length*8] = 0;
59             result[i+messages.length*8] |= SecToolMessageTransformation. →
                    ↪ ARGFLAG_REE_OPCODE_NSi_TXi;
60         }
61
62         return result;
63     }
64
65     /**
66     *
67     * @param messages
68     * @return
69     */
70     public static byte[] compileREE2ReaderMessages(int[] messages)
71         throws RuntimeException {
72
73         byte[] result = new byte[1];
74         result[0] = 0;
75
76         if(null == messages || messages.length < 1)
77             throw new RuntimeException("Oops!_compileREE2ReaderMessages_passed_argument_ →
                    ↪ verification_failed");
78
79         byte txBitMask = 1;
80         int count = 0;
81         for(int i=0; i<messages.length; i++) {
82
83             /* If FSM S2 then write result and shift txBitmask */
84             if((messages[i] & SecToolMessageTransformation.RETMASK_REE_FSM) == →
                    ↪ SecToolMessageTransformation.RETMASK_REE_FSM_S2) {
85
86                 /* If Tx = 1 then write 1 else 0 (default values are 0) */
87                 if((messages[i] & SecToolMessageTransformation.RETMASK_REE_TX) →
                    ↪ == 0x1)
88                     result[0] |= txBitMask;
89
90                 /* Shift txBitMask for next value */
91                 txBitMask <<= 1;
92                 count++;
93                 if(8 == count)
94                     break;
95             }
96         }
97
98         return result;
99     }
100 }
101 }

```

Izpis E.2: Primer opisa protokolnega analizatorja v programskem jeziku Java.

E.3 Izvedba vmesnika čitalca

```

1 package rpcbridge.reader;
2
3 public interface ReaderInterface {
4
5     /**
6     *
7     * @param message
8     * @throws RuntimeException
9     */
10    void write(byte[] message) throws RuntimeException;

```

```

11
12  /**
13   *
14   * @return
15   * @throws RuntimeException
16   */
17  byte[] read() throws RuntimeException;
18
19  /**
20   *
21   * @return
22   * @throws RuntimeException
23   */
24  int status() throws RuntimeException;
25
26  /**
27   *
28   */
29  void start();
30
31  }

```

Izpis E.3: Vmesnik čitalca.

```

1  package rpcbridge.reader;
2
3  import java.io.IOException;
4  import java.util.concurrent.TimeUnit;
5
6  import org.acplt.oncrpc.OncRpcException;
7  import org.apache.log4j.Level;
8  import org.apache.log4j.Logger;
9
10 public class ReaderAdapter extends Thread implements ReaderInterface {
11
12     static Logger logger = Logger.getLogger(ReaderAdapter.class);
13     private final int READ_POLL_TIMEOUT = 1100;
14     private ReaderBridgeBufServer server = null;
15
16     /**
17      *
18      * @param queue_length
19      */
20     public ReaderAdapter(int queue_length) {
21
22         try {
23             this.server = ReaderBridgeFactory.getReaderBridgeBufServer(queue_length, false);
24             logger.trace("Server_handle_created.");
25         } catch (RuntimeException e) {
26             e.printStackTrace();
27             logger.error("Oops!_Server_handle_creation_error._Exit");
28             System.exit(1);
29         }
30     }
31
32
33     /**
34      * ReaderInterface implementation
35      * Write to argQueue
36      */
37     @Override
38     public void write(byte[] message) {
39
40         try {
41             server.argQueue.put(message);
42         } catch (InterruptedException e) {
43             e.printStackTrace();
44             logger.error("Oops!_argQueue_put()_was_interrupted._Continue_to_ensure_ →
               ↪ secTool_error_handling.");

```

```

45         server.setStatusFlag(ReaderBridgeBufServer.DEVICE_WRITE_ERR);
46     }
47
48 }
49
50 /**
51  * ReaderInterface implementation
52  * Read from Queue
53  */
54 @Override
55 public byte[] read() {
56     byte[] retVal = null;
57
58     try {
59         retVal = server.retQueue.poll(READ_POLL_TIMEOUT, TimeUnit.MILLISECONDS);
60         if(retVal == null)
61             logger.trace("server_retQueue_poll()_timed_out");
62     } catch (InterruptedException e) {
63         e.printStackTrace();
64         logger.error("Oops!_argQueue_poll/take()_was_interrupted._Continue_to_ensure →
65             ↪ _secTool_error_handling.");
66         retVal = null;
67     }
68
69     return retVal;
70 }
71
72 /**
73  * ReaderInterface implementation
74  */
75 @Override
76 public int status() {
77     int result = 0;
78
79     result = server.getStatus();
80
81     return result;
82 }
83
84 /**
85  *
86  */
87 @Override
88 public void run() {
89     logger.setLevel(Level.INFO);
90     try {
91         this.server.run();
92     } catch (OnCRpcException e) {
93         e.printStackTrace();
94         logger.error("Oops!_Server_handle_creation_error._Exit.");
95         System.exit(1);
96     } catch (IOException e) {
97         e.printStackTrace();
98         logger.error("Oops!_Server_handle_creation_error._Exit.");
99         System.exit(1);
100     }
101 }
102
103 }
104
105 }

```

Seznam strojne in programske opreme

F.1 Strojna oprema za razvoj emulacijskega sistema

Računalnik PC z nameščenim operacijskim sistemom Windows 7 vključuje naslednjo programsko in strojno opremo:

- Aten UC-232A: pretvornik USB - RS232,
- Xilinx ISE 10.1.03,
- Xilinx EDK 10.1.03.

F.2 Programska oprema za razvoj emulacijskega sistema

Računalnik CentOS 5.3 x86 (virtualna slika) vključuje naslednjo programsko in strojno opremo:

- PetaLinux 0.4 (uClinux) razvojni sistem
- Knjižnice in podporni programi za grajenje:

F.3 Strojna oprema emulacijskega sistema

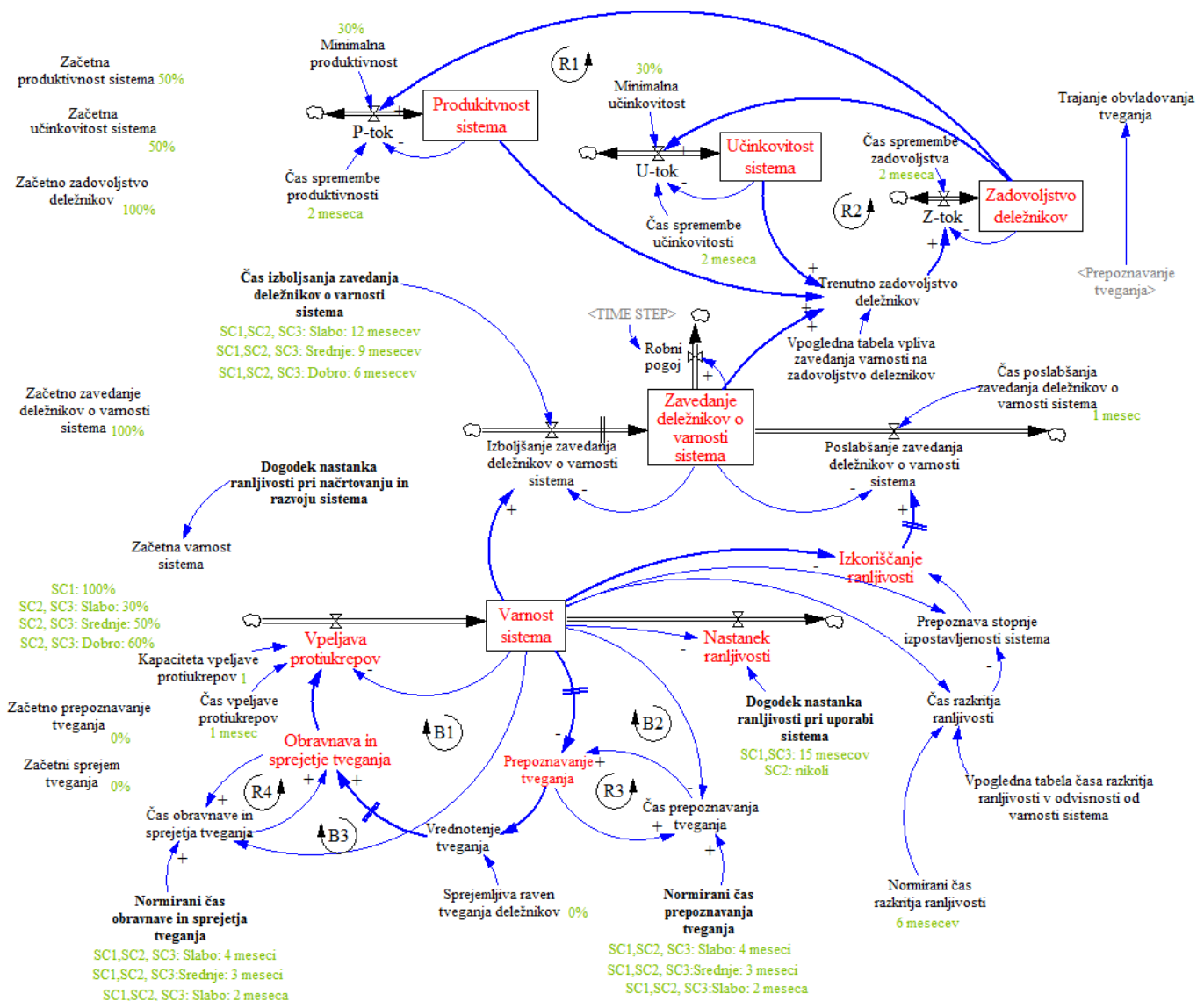
- Xilinx University Program Virtex-II Pro Development System,
- Kingston KVR266X64C25/256 256 MB DDR pomnilnik RAM,
- USB/JTAG priključek.

F.4 Programska oprema emulacijskega sistema

- Programirljiv računalnik MicroBlaze z omogočenim podatkovnim in ukaznim predpomnilnikom, časovnikom in modulom za rokovanje s prekinitvami.
- Programirljiva naprave: Ethernet, UART, vmesnik DDR RAM in GPIO.
- Slika PetaLinux 0.4 (uClinux) z zagonskim nalagalnikom in gonilniki za programirljivo strojno nastavitev ciljnega sistema.

G

Simulacijski model



Slika G.1: Podrobni model medsebojnega vpliva karakteristik kakovosti sistema.

```

1  Cas Izboljsanja Zavedanja Deleznikov o Varnosti Sistema=
2      12
3      Units: Month [0.5,24,0.1]
4
5  Cas Obravnave in Sprejetja Tveganja=
6      IF THEN ELSE( (1-Varnost Sistema)<Obravnava in Sprejetje Tveganja, Normiran Cas Obravnave →
7          ↪ in Sprejetja Tveganja
8          *0.01, Normiran Cas Obravnave in Sprejetja Tveganja
9      )
10     Units: Month
11
12  Cas Poslabsanja Zavedanja Deleznikov o Varnosti Sistema=
13      1
14     Units: Month [0.5,24,0.1]
15
16  Cas Prepoznavanja Tveganja=
17      IF THEN ELSE( (1-Varnost Sistema)<Prepoznavanje Tveganja, Normiran Cas Prepoznavanja →
18          ↪ Tveganja
19          *0.001, Normiran Cas Prepoznavanja Tveganja

```

```

18      )
19      Units: Month [0,?]
20
21      Cas Razkritja Ranljivosti=
22          Vpogledna Tabela Casa Razkritja Ranljivosti v odvisnosti od Varnosti Sistema
23          (Varnost Sistema)*Normiran Cas Razkritja Ranljivosti
24      Units: Month [0,?]
25
26      Cas Spremembe Produktivnosti=
27          2
28      Units: Month [0.5,12,0.1]
29
30      Cas Spremembe Ucinkovitosti=
31          2
32      Units: Month [0.5,12,0.1]
33
34      Cas Spremembe Zadovoljstva=
35          2
36      Units: Month [0.5,12,0.1]
37
38      Cas Vpeljave=
39          1
40      Units: Month
41
42      Dogodek Nastanka Ranljivosti pri Snovanju Sistema=
43          1
44      Units: Dmnl [0,1,0.05]
45
46      Dogodek Nastanka Ranljivosti pri Uporabi Sistema=
47          150
48      Units: Month [0,100,0.5]
49
50      FINAL TIME = 100
51          Units: Month
52          The final time for the simulation.
53
54      INITIAL TIME = 0
55          Units: Month
56          The initial time for the simulation.
57
58      izboljsanje zavedanja deleznikov o varnosti sistema=
59          max(DELAY3I(Varnost Sistema, 3*Cas Izboljsanja Zavedanja Deleznikov o Varnosti Sistema
60      /4,0)-Zavedanje Deleznikov o Varnosti Sistema,0)/(Cas Izboljsanja Zavedanja Deleznikov o →
61          ↪ Varnosti Sistema
62          /4)
63      Units: 1/Month
64      max(DELAY3I(Realna Varnost Sistema, 3*Cas Pozitivne Spremembe
65          Zaupanja v Izboljsano Varnost Sistem/4,0)-Zavedanje Varnosti
66          Sistema Interesnih Skupin,0)/(Cas Pozitivne Spremembe Zaupanja v
67          Izboljsano Varnost Sistem /4)
68
69      Izkoriscanje Ranljivosti=
70          min((1-Varnost Sistema), Prepoznava Stopnje Izpostavljenosti Sistema)
71      Units: Dmnl [0,?]
72
73      Kapaciteta Vpeljave=
74          1
75      Units: Dmnl [0,1,0.1]
76
77      Minimalna Produktivnost=
78          0.3
79      Units: Dmnl [0,1,0.1]
80
81      Minimalna Učinkovitost=
82          0.3
83      Units: Dmnl [0,1,0.1]
84
85      nastanek ranljivosti=
86          IF THEN ELSE(Varnost Sistema>=0, 1*PULSE(Dogodek Nastanka Ranljivosti pri Uporabi →
87          ↪ Sistema

```

```

86 , 1), 0)
87 Units: 1/Month
88
89 Normiran Cas Obravnave in Sprejetja Tveganja=
90 4
91 Units: Month
92
93 Normiran Cas Prepoznavanja Tveganja=
94 4
95 Units: Month [0,?]
96
97 Normiran Cas Razkritja Ranljivosti=
98 6
99 Units: Month
100
101 Obravnava in Sprejetje Tveganja= DELAY INFORMATION (
102 Vrednotenje Tveganja, Cas Obravnave in Sprejetja Tveganja,ZACETNI Sprejem Tveganja
103 )
104 Units: Dmnl
105
106 poslabsanje zavedanja deleznikov o varnosti sistema=
107 max((Zavedanje Deleznikov o Varnosti Sistema*Izkoriscanje Ranljivosti)/Cas Poslabsanja →
108 ↪ Zavedanja Deleznikov o Varnosti Sistema
109 ,0)
110 Units: 1/Month [0,?]
111
112 Prepознава Stopnje Izpostavljenosti Sistema= DELAY INFORMATION (
113 max(1-Varnost Sistema,0),Cas Razkritja Ranljivosti ,0)
114 Units: Dmnl [0,?]
115
116 Prepознаvanje Tveganja= DELAY INFORMATION (
117 1-Varnost Sistema, Cas Prepознаvanja Tveganja,ZACETNO Prepознаvanje Tveganja
118 )
119 Units: Dmnl
120
121 Produktivnost Sistema= INTEG (
122 ptok,
123 ZACETNA Produktivnost Sistema)
124 Units: Dmnl [0,1.05]
125
126 ptok=
127 IF THEN ELSE( Produktivnost Sistema>1, min(Zadovoljstvo Deleznikov-0.7,0)
128 , IF THEN ELSE(Produktivnost Sistema<=Minimalna Produktivnost
129 , max(Zadovoljstvo Deleznikov-0.7,0), Zadovoljstvo Deleznikov-0.7))/Cas Spremembe →
130 ↪ Produktivnosti
131 Units: 1/Month
132
133 robni pogoj=
134 IF THEN ELSE(Zavedanje Deleznikov o Varnosti Sistema>1, Zavedanje Deleznikov o Varnosti →
135 ↪ Sistema
136 -1, 0)/TIME STEP
137 Units: 1/Month [0,?]
138
139 SAVEPER =
140 TIME STEP
141 Units: Month [0,?]
142 The frequency with which output is stored.
143
144 Sprejemljiva Raven Tveganja Deleznikov=
145 0
146 Units: Dmnl [0,1,0.1]
147
148 TIME STEP = 0.03125
149 Units: Month [0,?]
150 The time step for the simulation.
151
152 Trajanje Nacrtovanja Ukrepanja Preverjanja in Ukrepanja pri Varovanju Informacij
153 =
154 IF THEN ELSE( Prepознаvanje Tveganja>0.1, 1, 0)
155 Units: Dmnl [0,?]

```

```

153
154 Trenutno Zadovoljstvo Deleznikov=
155     (Produktivnost Sistema-0.5)/4+(Ucinkovitost Sistema-0.5)/4+Vpliv Zavedanja Varnosti na →
156     ↪ Zadovoljstvo Deleznikov
157 (Zavedanje Deleznikov o Varnosti Sistema)*2/4
158 Units: Dmnl
159
160 Ucinkovitost Sistema= INTEG (
161     utok,
162     ZACETNA Ucinkovitost Sistema)
163 Units: Dmnl [0,1.05]
164
165 utok=
166     IF THEN ELSE( Ucinkovitost Sistema>=1, min(Zadovoljstvo Deleznikov-0.7,0),
167     IF THEN ELSE(Ucinkovitost Sistema<=Minimalna UĀtinkovitost,
168     max(Zadovoljstvo Deleznikov-0.7,0), Zadovoljstvo Deleznikov-0.7))/Cas Spremembe →
169     ↪ Ucinkovitosti
170 Units: 1/Month
171
172 Varnost Sistema= INTEG (
173     vpeljava protiukrepov-nastanek ranljivosti,
174     ZACETNA Varnost Sistema)
175 Units: Dmnl [0,1.05]
176
177 vpeljava protiukrepov=
178     IF THEN ELSE(Varnost Sistema<1, Obravnava in Sprejetje Tveganja*Kapaciteta Vpeljave
179     , 0)/Cas Vpeljave
180 Units: 1/Month
181
182 Vpliv Zavedanja Varnosti na Zadovoljstvo Deleznikov(
183     [(0,-1)-(2,1)], (0,-1), (0.1,-1), (0.2,-1), (0.3,-1), (0.4,-1), (0.5,-1), (0.6,-1
184     ), (0.649412,-0.978648), (0.677647,-0.928826), (0.7,-0.84023), (0.71,-0.51387),
185     (0.72,-0.099775), (0.73,0.20691), (0.74,0.38153), (0.75,0.49285), (0.76,0.57), (
186     0.77,0.62666), (0.78,0.67006), (0.79,0.70438), (0.8,0.73221), (0.81,0.75524), (0.82
187     ,0.77461), (0.83,0.79113), (0.84,0.80539), (0.85,0.81782), (0.86,0.82876), (0.87
188     ,0.83846), (0.88,0.84711), (0.89,0.85489), (0.9,0.86191), (0.91,0.86828), (0.92,
189     0.87409), (0.93,0.87941), (0.94,0.8843), (0.95,0.88881), (0.96,0.89297), (0.97,0.89684
190     ), (0.98,0.90044), (0.99,0.90379), (1,0.90693), (1.01,0.90987), (1.02,0.91262), (
191     1.03,0.91522), (1.04,0.91766), (1.05,0.91997), (1.06,0.92215), (1.07,0.92422), (
192     1.08,0.92617), (1.09,0.92804), (1.1,0.9298), (1.11,0.93149), (1.12,0.93309), (1.13
193     ,0.93462), (1.14,0.93609), (1.15,0.93749), (1.16,0.93882), (1.17,0.94011), (1.18
194     ,0.94134), (1.19,0.94252), (1.2,0.94365), (1.21,0.94474), (1.22,0.94579), (1.23,
195     0.9468), (1.24,0.94777), (1.25,0.94871), (1.26,0.94961), (1.27,0.95049), (1.28,0.95133
196     ), (1.29,0.95215), (1.3,0.95293), (1.31,0.9537), (1.32,0.95444), (1.33,0.95515),
197     (1.34,0.95584), (1.35,0.95652), (1.36,0.95717), (1.37,0.9578), (1.38,0.95842), (
198     1.39,0.95901), (1.4,0.95959), (1.41,0.96016), (1.42,0.9607), (1.43,0.96124), (1.44
199     ,0.96176), (1.45,0.96226), (1.46,0.96275), (1.47,0.96323), (1.48,0.9637), (1.49,
200     0.96415), (1.5,0.9646), (1.51,0.96503), (1.52,0.96546), (1.53,0.96587), (1.54,0.96627
201     ), (1.55,0.96666), (1.56,0.96705), (1.57,0.96742), (1.58,0.96779), (1.59,0.96815
202     ), (1.6,0.9685), (1.61,0.96885), (1.62,0.96918), (1.63,0.96951), (1.64,0.96983),
203     (1.65,0.97015), (1.66,0.97046), (1.67,0.97076), (1.68,0.97105), (1.69,0.97135),
204     (1.7,0.97163), (1.71,0.97191), (1.72,0.97218), (1.73,0.97245), (1.74,0.97271), (
205     1.75,0.97297), (1.76,0.97323), (1.77,0.97347), (1.78,0.97372), (1.79,0.97396), (
206     1.8,0.97419), (1.81,0.97442), (1.82,0.97465), (1.83,0.97487), (1.84,0.97509), (1.85
207     ,0.97531), (1.86,0.97552), (1.87,0.97573), (1.88,0.97593), (1.89,0.97613), (1.9,
208     0.97633), (1.91,0.97653), (1.92,0.97672), (1.93,0.9769), (1.94,0.97709), (1.95,0.97727
209     ), (1.96,0.97745), (1.97,0.97763), (1.98,0.9778), (1.99,0.97797), (2,0.97814))
210 Units: Dmnl
211
212 [(0,-1)-(10,10)], (0,-1), (0.5,-1), (0.7,-1), (0.75,-0.5), (0.8,0.5), (
213     0.85,0.75), (0.9,0.8), (0.95,0.85), (0.975,0.9), (0.98,0.95), (0.99,0.
214     99), (1,1), (1.1,1)
215
216 Vpogledna Tabela Casa Razkritja Ranljivosti v odvisnosti od Varnosti Sistema
217 (
218     [(0,0)-(1.1,1)], (0,0.99686), (0.985,0.99682), (0.9851,0.99681), (0.9852,0.9968
219     ), (0.9853,0.99679), (0.9854,0.99677), (0.9855,0.99675), (0.9856,0.99673), (0.9857
220     ,0.9967), (0.9858,0.99666), (0.9859,0.99661), (0.986,0.99656), (0.9861,0.99649)
221     , (0.9862,0.99641), (0.9863,0.99631), (0.9864,0.99619), (0.9865,0.99604), (0.9866
222     ,0.99586), (0.9867,0.99564), (0.9868,0.99537), (0.9869,0.99504), (0.987,0.99464
223     ), (0.9871,0.99415), (0.9872,0.99356), (0.9873,0.99283), (0.9874,0.99194), (0.9875

```

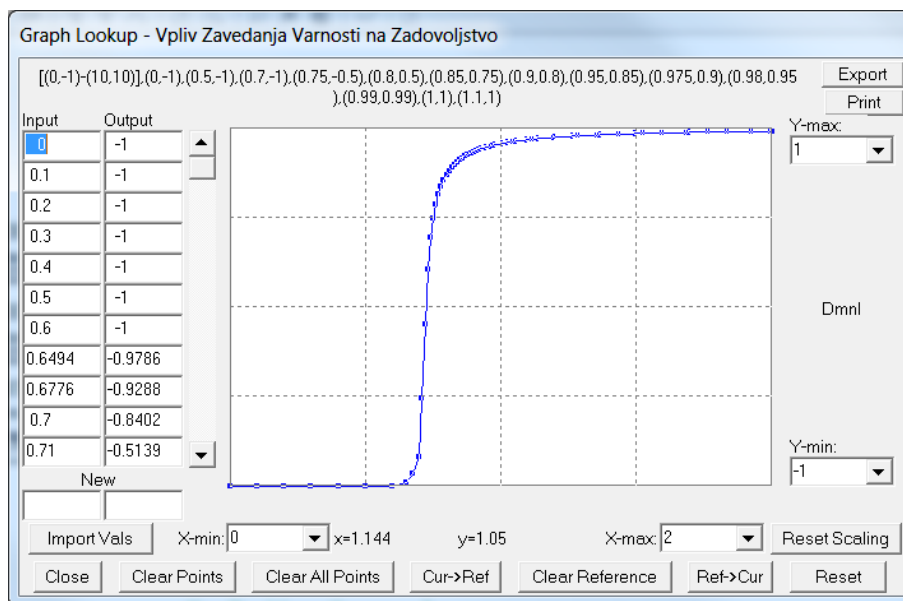
```

221 ,0.99086),(0.9876,0.98954),(0.9877,0.98794),(0.9878,0.98598),(0.9879,0.98361
222 ),(0.988,0.98073),(0.9881,0.97724),(0.9882,0.97301),(0.9883,0.9679),(0.9884
223 ,0.96173),(0.9885,0.95433),(0.9886,0.94545),(0.9887,0.93485),(0.9888,0.92227
224 ),(0.9889,0.9074),(0.989,0.88995),(0.9891,0.86964),(0.9892,0.8462),(0.9893,
225 0.81945),(0.9894,0.78926),(0.9895,0.75566),(0.9896,0.71881),(0.9897,0.67906
226 ),(0.9898,0.63694),(0.9899,0.59312),(0.99,0.54843),(0.9901,0.50374),(0.9902
227 ,0.45992),(0.9903,0.4178),(0.9904,0.37805),(0.9905,0.3412),(0.9906,0.3076),
228 (0.9907,0.27741),(0.9908,0.25066),(0.9909,0.22722),(0.991,0.20691),(0.9911,
229 0.18946),(0.9912,0.17459),(0.9913,0.16201),(0.9914,0.15141),(0.9915,0.14253
230 ),(0.9916,0.13513),(0.9917,0.12896),(0.9918,0.12385),(0.9919,0.11962),(0.992
231 ,0.11613),(0.9921,0.11325),(0.9922,0.11088),(0.9923,0.10893),(0.9924,0.10732
232 ),(0.9925,0.106),(0.9926,0.10492),(0.9927,0.10403),(0.9928,0.1033),(0.9929,
233 0.10271),(0.993,0.10222),(0.9931,0.10182),(0.9932,0.10149),(0.9933,0.10122)
234 ,(0.9934,0.101),(0.9935,0.10082),(0.9936,0.10067),(0.9937,0.10055),(0.9938,
235 0.10045),(0.9939,0.10037),(0.994,0.1003),(0.9941,0.10025),(0.9942,0.1002),(
236 0.9943,0.10017),(0.9944,0.10014),(0.9945,0.10011),(0.9946,0.10009),(0.9947,
237 0.10007),(0.9948,0.10006),(0.9949,0.10005),(0.995,0.10004),(1.1,0.1))
238 Units: Dmnl
239 x=0.985:0.0001:0.995; x=[0,x,1.1];
240 f='(1-tanh((t-0.99)/0.001))/2/1.115+0.1';
241 vensimGraphData(x,eval(vectorize(f)))
242
243 Vrednotenje Tveganja=
244 max(Prepoznavanje Tveganja-Sprejemljiva Raven Tveganja Deleznikov,0)
245 Units: Dmnl
246
247 ZACETNA Produktivnost Sistema=
248 0.5
249 Units: Dmnl [0,1,0.1]
250
251 ZACETNA Ucinovitost Sistema=
252 0.5
253 Units: Dmnl [0,1,0.1]
254
255 ZACETNA Varnost Sistema=
256 Dogodek Nastanka Ranljivosti pri Snovanju Sistema
257 Units: Dmnl [0,?]
258
259 ZACETNI Sprejem Tveganja=
260 0
261 Units: Dmnl
262
263 ZACETNO Prepoznavanje Tveganja=
264 0
265 Units: Dmnl
266
267 ZACETNO Zadovoljstvo Deleznikov=
268 1
269 Units: Dmnl [0,1,0.1]
270
271 ZACETNO Zavedanje Deleznikov o Varnosti Sistema=
272 1
273 Units: Dmnl [0,1,0.1]
274
275 Zadovoljstvo Deleznikov= INTEG (
276 ztok,
277 ZACETNO Zadovoljstvo Deleznikov)
278 Units: Dmnl [0,1.05]
279
280 Zavedanje Deleznikov o Varnosti Sistema= INTEG (
281 -robni pogoj-poslabljanje zavedanja deleznikov o varnosti sistema+izboljsanje zavedanja →
282 ↵ deleznikov o varnosti sistema
283 ,
284 ZACETNO Zavedanje Deleznikov o Varnosti Sistema)
285 Units: Dmnl [0,1.05]
286
287 ztok=
288 IF THEN ELSE(Zadovoljstvo Deleznikov>=1, min(Trenutno Zadovoljstvo Deleznikov
289 ,0), IF THEN ELSE(Zadovoljstvo Deleznikov<=0.01, max(Trenutno Zadovoljstvo Deleznikov
290 ,0), Trenutno Zadovoljstvo Deleznikov)

```

290)/Cas Spremembe Zadovoljstva
291 Units: 1/Month

Izpis G.1: Vmesnik čitalca.



Slika G.2: Vpogledna tabela Vpliv zavedanja varnosti na zadovoljstvo.

H

Izvorna koda

Izvorna koda je dostopna na optičnemu nosilcu DVD.

2.1	Diagram primera uporabe domenskega modela z razvrstitvijo legitimnih storitev.	5
2.2	Razredni diagram domenskega arhitekturnega modela.	5
2.3	Diagram aktivnosti prikazuje tok podatkov domenskega arhitekturnega modela in pripadajoče fizične razmejitve.	5
2.4	Diagram vzroka in posledic karakteristik kakovosti sistema.	7
2.5	Proces obvladovanja tveganja za varovanje informacij ISO/IEC 27005:2008.	13
3.1	Diagram primera uporabe z razvrstitvijo napadov in legitimnih storitev.	18
3.2	Domenski arhitekturni model orodja.	22
3.3	Bločni diagram načina preizkušanja z emulacijo.	22
3.4	Arhitekturni model orodja.	25
3.5	Generična struktura vezja RFID.	26
3.6	Emulacijsko okolje in binarni aplikacijski vmesnik.	26
3.7	Diagram stanja značke.	28
3.8	Binarni aplikacijski vmesnik demonstracijskega sistema.	29
3.9	Diagram strukture strežnika in odjemalca komunikacijskega mostu čitalca.	32
3.10	Bločni diagram sistema MicroBlaze z dodanim emulacijskim okoljem.	35
3.11	Diagram strukture strežnika in odjemalca komunikacijskega tunela čitalca.	37

3.12	Diagram strukture orodja za preizkušanje varnosti.	38
3.13	Grafični vmesnik orodja za preizkušanje varnosti.	40
3.14	Diagram poteka transformacije sporočila čitalca.	41
3.15	Bločni diagram notranje strukture čitalca.	43
3.16	Diagram postavitve orodja.	46
3.17	Prikaz delovanja spletne strani sistema za prikaz dogodkov.	47
3.18	Konceptualni podatkovni model.	47
3.19	Logični podatkovni model.	48
3.20	Diagram aktivnosti zajetja dogodkov.	51
3.21	Diagram zaporedja komunikacije sistema in uporabe orodja.	52
3.22	Postopek vnosa sporočila čitalca.	54
3.23	Postopek spremembe sporočila čitalca.	55
3.24	Postopek izbrisa sporočila čitalca.	56
3.25	Postopek spremembe stanja značke.	57
3.26	Postopek vnosa sporočila značke.	58
3.27	Postopek spremembe sporočila značke.	59
3.28	Postopek izbrisa sporočila značke.	60
3.29	Diagram poteka metode za zagotavljanje varnosti za okolja omrežnih sistemov radio-frekvenčne identifikacije.	62
4.1	Tipologija lastnosti porazdeljenih sistemov.	68
4.2	Omrežna arhitektura orodja.	73
4.3	Model medsebojnega vpliva karakteristik kakovosti sistema.	78
4.4	Scenarij “Nova ranljivost pri aktivni uporabi sistema” – Slaba kakovost obvladovanja tveganja.	80
4.5	Scenarij “Naknadno odkrita ranljivost” – Srednja kakovost obvladovanja tveganja.	82
4.6	Scenarij “Nova ranljivost pri aktivni uporabi sistema” – Dobra kakovost obvladovanja tveganja.	84

4.7	Primerjava obnašanja spremenljivk sistema treh izidov scenarija “Nova ranljivost pri aktivni uporabi sistema”.	85
4.8	Scenarij “Nabava ranljivega sistema” — Slaba kakovost obvladovanja tveganja.	88
4.9	Scenarij “Nabava ranljivega sistema” – Srednja kakovost obvladovanja tveganja.	89
4.10	Scenarij “Nabava ranljivega sistema” – Dobra kakovost obvladovanja tveganja.	92
4.11	Primerjava obnašanja spremenljivk sistema treh izidov scenarija “Nabava ranljivega sistema”.	93
4.12	Scenarij “Nabava ranljivega sistema z nastankom nove ranljivosti” – Slaba kakovost obvladovanja tveganja.	96
4.13	Scenarij “Nabava ranljivega sistema z nastankom nove ranljivosti” — Srednja kakovost obvladovanja tveganja.	98
4.14	Scenarij “Nabava ranljivega sistema z nastankom nove ranljivosti” — Dobra kakovost obvladovanja tveganja.	101
4.15	Primerjava obnašanja spremenljivk sistema treh izidov scenarija “Nabava ranljivega sistema z nastankom nove ranljivosti”.	102
A.1	Diagram postavitve uporabljene strojne in programske opreme	108
A.2	Diagram postavitve UML: Definicija in gradnja strojne opreme ter osnovnih orodij za prevajanje in gradnjo programske opreme za ciljni sistem v razvojnem okolju Xilinx EDK	109
A.3	Diagram postavitve UML: <i>Namestitve</i> bitne slike na programirljivo vezje FPGA preko priključka USB/JTAG in vzpostavitev strojne opreme ciljnega sistema	109
A.4	Diagram postavitve UML: Vzpostavljena strojna oprema ciljnega sistema na programirljivem vezju FPGA	110
A.5	Diagram postavitve UML: Definicija in gradnja operacijskega sistema uClinux z orodji Petalinux in kreacija datotečnega sistema ter uporabniških programov. Postopek gradnje operacijskega sistema <i>uporablja</i> specifikacijo strojne opreme ciljnega sistema in <i>uporablja</i> orodja za gradnjo in prevajanje izvirne kode. Ta oprema je bila pridobljena v prvem koraku (glej diagram A.2). Datotečni sistem ciljnega sistema vsebuje dodatno aplikacijo za testiranje omrežne poveztivosti in strojne opreme ciljnega sistema .	111

A.6	Diagram postavitve UML: <i>Namestitev</i> slike celotne programske opreme na ciljni sistem preko priključka USB/JTAG z uporabo razhroščevalne enote procesorja MicroBlaze.	112
A.7	Diagram postavitve UML: Nameščen in delujoč ciljni sistem. Prijava v ciljni sistem je sprva mogoča preko serijskega terminala z uporabo programa PuTTY ali minicom. Če je mrežna naprava ciljnega sistema nastavljena pravilno, se je mogoče prijaviti tudi z oddaljenim dostopom z uporabo odjemalca Telnet	113
A.8	Diagram postavitve UML: Test oddaljene aplikacije	114
B.1	Kreacija novega projekta s pomočjo čarovnika v programskem orodju EDK in priključitev repozitorija strojne opreme.	115
B.2	Kreacija novega načrta.	116
B.3	Določitev razvojnega sistema (za XUP Virtex-II Pro Dev System se zahteva izpolnjen prvi korak - uspešna priključitev repozitorija s slike B.1).	116
B.4	Izbira programirljivega procesorja MicroBlaze.	117
B.5	Določitev osnovnih programirljivih parametrov za kreacijo procesorja MicroBlaze.	117
B.6	Izbira in nastavitve dodatnih vhodno/izhodnih naprav (1).	118
B.7	Izbira in nastavitve dodatnih vhodno/izhodnih naprav (2).	118
B.8	Izbira in nastavitve dodatnih vhodno/izhodnih naprav (3).	119
B.9	Nastavitve predpomnilnika procesorja MicroBlaze.	119
B.10	Izbira privzete vhodno/izhodne naprave, izbira zagonskega pomnilnika in testnih programov.	120
B.11	Določitev parametrov testnega programa.	120
B.12	Povzetek nastavitve strojne opreme.	121
B.13	Povzetek nastavitve projekta.	121
B.14	Projekt: Vidik sestave strojne opreme (1): Določitev vmesnikov vodila računalnika.	122
B.15	Projekt: Vidik sestave strojne opreme (2): Priključitev naprav na vhodno/izhodna vrata.	123

B.16 Projekt: Vidik sestave strojne opreme (3): Določitev pomnilniških naslovov naprav (<i>Pomnilnik RAM se nahaja na razponu naslovov 0x90000000 – 0x9FFFFFFF</i>).	124
B.17 Projekt: Grafični prikaz organizacije strojne opreme.	125
B.18 Fina nastavitvev krmilnika RAM dostopna s slike B.14 (<i>Določitev parametrov pomnilnika RAM Kingston KVR266X64C25/256</i>).	126
B.19 Gradnja strojne opreme: Sinteza, implementacija in sestava bitne slike strojne opreme (<i>Generate Netlist in Generate Bitstream</i>).	127
B.20 Sprememba vnosa datoteke <i>system.mss</i> : Sprememba definicije operacijskega sistema (<i>vnos je mogoče spremeniti tudi preko grafičnega vmesnika v Software:Software Platform Settings:Software Platform in Software:Software Platform Settings:OS and Libraries</i>).129	129
B.21 Gradnja osnovne programske opreme z lastnimi določili za operacijski sistem (BSP): Gradnja knjižnic in vseh uporabniških aplikacij (<i>Generate Libraries and BSPs, Build All User Application</i>).	130
B.22 Pred nadaljevanjem se je najbolje prepričati ali se je postopek gradnje strojne opreme in programskih orodij pravilno zaključil. Za ta namen se posodobi bitna slika strojne in programske opreme (<i>Update Bitstream</i>) ter namesti na sistem FPGA z uporabo povezave USB/JTAG (<i>Download bitstream</i>).	131
B.23 V programskih orodjih je bil določen standardni vhod/izhod UART, zato se za dostop do testnega sistema uporabi serijsko vodilo (<i>v našem primeru je dostop posreden na vratih COM7 z uporabo adapterja USB/RS232 Aten UC-232A</i>).	131
B.24 Za dostop se uporabi odjemalec PuTTY.	132
B.25 Z malo sreče ☺ se na serijskem terminalu izpiše besedilo uspešnega testiranja pomnilnika DDR RAM (<i>sicer poskusite ponovno ponastaviti sistem (RESET)</i>). To je tudi eden izmed <i>potrebnih pogojev</i> za uspešno namestitve slike operacijskega sistema in datotečne sistema, ki se bosta nahajala v pomnilniku RAM.	132
B.26 Zadnji korak je sprememba izbire aktivnega programa za neskončno zanko. Operacijski sistem se bo v nadaljevanju naložil preko razhroščevalnega vmesnika MicroBlaze z uporabo povezave USB/JTAG.	133
C.1 Izbira projekta (1).	137
C.2 Izbira projekta (2).	138
C.3 Izbira nastavitvev jedra operacijskega sistema (1).	139

C.4	Izbira nastavitev jedra operacijskega sistema (2).	140
C.5	Izbira nastavitev jedra operacijskega sistema (3).	140
C.6	Izbira nastavitev jedra operacijskega sistema (4).	141
C.7	Izbira nastavitev jedra operacijskega sistema (5).	141
C.8	Izbira nastavitev jedra operacijskega sistema (6).	142
C.9	Izbira nastavitev jedra operacijskega sistema (7).	142
C.10	Izbira nastavitev jedra operacijskega sistema (8).	143
C.11	Izbira nastavitev jedra operacijskega sistema (9).	143
C.12	Izbira nastavitev jedra operacijskega sistema (10).	144
C.13	Izbira nastavitev uporabniške programske opreme (1).	144
C.14	Izbira nastavitev uporabniške programske opreme (2).	145
C.15	Namestitev bitne slike z izbrano aplikacijo <i>bootloop</i> na sistem FPGA z uporabo povezave USB/JTAG (<i>Download bitstream</i>).	147
C.16	Zagon razhroščevalne konzole (<i>Launch XMD</i>) in samodejna povezava na razhroščevalno enoto procesorja MicroBlaze.	148
C.17	Nalaganje slike operacijskega in datotečnega sistema <code>image.bin</code> v pomnilnik RAM ciljnega sistema z začetnim naslovom <code>0x90000000</code> (<i>proces nalaganja je dolgotrajen, privoščite si kavo</i>), ponastavitev procesorja in nastavitve ukaznega kazalca na naslov <code>0x90000000</code> .	149
C.18	Z malo sreče se izpiše zagonska sekvenca operacijskega sistema uClinux.	150
C.19	Zagonska sekvenca se konča s prijavo v sistem. Privzeto geslo uporabnika <i>root</i> je <i>root</i> .	150
C.20	V primeru, da je ciljni sistem v omrežju s storitvijo <i>DHCP</i> se nastavitve mrežne naprave izvede z uporabo programa <i>dhcpcd</i> . Uporaba znaka <i>ℳ</i> pošlje proces v ozadje. Pridobljeni naslov IP se izpiše z uporabo ukaza <i>ifconfig</i> . V primeru, da storitev <i>DHCP</i> ni na voljo, se mrežno napravo nastavi z uporabo ukaza <i>ifconfig</i> .	151
C.21	uClinux ima privzeto zagnane naslednje mrežne storitve: Portmap, HTTP, FTP in Telnet.	152
C.22	Z uporabo odjemalca Telnet (program PuTTY) je mogoč oddaljen dostop do ciljnega sistema, če se le oba nahajata znotraj iste omrežne skupine.	153
C.23	Z malo sreče ☺ se oddaljena prijava v ciljni sistem prav tako uspešno zaključi.	154

C.24	Preizkus naprav GPIO s programom <i>gpio-test</i> . Posledica prvih dveh ukazov so prižgane diode LED in detekcija nizkega logičnega stanja stikala C.25, posledica drugih dveh pa ugasnjene diode LED in detekcija visokega logičnega stanja stikala C.26.	154
C.25	Test razvojnega sistema FPGA (1).	155
C.26	Test razvojnega sistema FPGA (2).	155
D.1	Nastavitev mrežne naprave in kreacija procesa strežnika RPC testne aplikacije na ciljnem sistemu.	165
D.2	Klic oddaljene procedure s sistema CentOS.	165
D.3	Preverjanje pravilnosti izvedbe programa RPC testne aplikacije na ciljnem sistemu.	166
G.1	Podrobni model medsebojnega vpliva karakteristik kakovosti sistema. . . .	176
G.2	Vpogledna tabela Vpliv zavedanja varnosti na zadovoljstvo.	182

Tabele

2.1	Tabela groženj, napadov in učinkov po tipologiji namernih groženj ISO/IEC 27005:2008.	11
4.1	Uporabniška transparentnost pri zasnovi, razvoju in gradnji orodja.	69
4.2	Uporabniška transparentnost pri preizkušanju varnosti.	69
4.3	Privzeti začetni pogoji modela.	77
4.4	Sprememba začetnih vrednosti simulacije za scenarij "Nova ranljivost pri aktivni uporabi sistema" – Srednja kakovost obvladovanja tveganja.	81
4.5	Začetni pogoji modela za scenarij "Nova ranljivost pri aktivni uporabi sistema" – Dobra kakovost obvladovanja tveganja.	83
4.6	Privzeti začetni pogoji modela za scenarij "Nabava ranljivega sistema".	86
4.7	Začetni pogoji modela za scenarij "Nabava ranljivega sistema" — Srednja kakovost obvladovanja tveganja.	87
4.8	Začetni pogoji modela za scenarij "Nabava ranljivega sistema" – Dobra kakovost obvladovanja tveganja.	90
4.9	Privzeti začetni pogoji modela za scenarij "Nabava ranljivega sistema z nastankom nove ranljivosti".	94
4.10	Začetni pogoji modela za scenarij "Nabava ranljivega sistema z nastankom nove ranljivosti" — Srednja kakovost obvladovanja tveganja.	97
4.11	Začetni pogoji modela za scenarij "Nabava ranljivega sistema z nastankom nove ranljivosti" — Dobra kakovost obvladovanja tveganja.	99

Izpisi

3.1	Opis protokola Komunikacijskega tunela značke.	30
3.2	Opis protokola Komunikacijski tunel čitalca v jeziku RPCL.	36
3.3	Opis spletne storitve za prenos dogodka elektronske kode v jeziku WSDL.	44
3.4	Poizvedbe SQL uporabljene pri zajetju in prikazu dogodkov.	49
B.1	UKAZNA LUPINA - Vsebina repozitorija.	128
C.1	UKAZNA LUPINA - Razpakiranje arhiva uClinux in orodij Petalinux ter kreacija skripte za hitri dostop do razvojnega okolja Petalinux.	135
C.2	UKAZNA LUPINA - Namestitev osnovnih knjižnic za gradnjo.	136
C.3	UKAZNA LUPINA - Izbira strojne platforme in projekta za gradnjo operacijskega sistema.	137
C.4	UKAZNA LUPINA - Izbira strojne platforme in projekta za gradnjo operacijskega sistema.	138
C.5	UKAZNA LUPINA - Izbira strojne platforme in projekta za gradnjo operacijskega sistema.	139
C.6	UKAZNA LUPINA - Gradnja operacijskega sistema.	146
C.7	UKAZNA LUPINA - Prekopiranje slike v mapo projekta EDK	147
D.1	UKAZNA LUPINA - Kreacija projekta testne aplikacije.	157
D.2	Vsebina datoteke EmbeddedTestApp1.c.	157
D.3	Vsebina datoteke msg_svc.c.	158
D.4	Vsebina datoteke msg.h.	159
D.5	Vsebina datoteke Makefile.	160

D.6	UKAZNA LUPINA - Gradnja testne aplikacije in namestitvev v datotečni sistem.	161
D.7	UKAZNA LUPINA - Kopiranje datoteke protocols v datotečni sistem ciljnega sistema. . . .	161
D.8	UKAZNA LUPINA - Gradnja uporabniške programske opreme in posodobitev binarne slike operacijskega in datotečnega sistema.	161
D.9	UKAZNA LUPINA - Kreacija projekta odjemalca za testno aplikacijo.	162
D.10	Vsebina datoteke EmbeddedTestApp1Client.c.	162
D.11	Vsebina datoteke msg_clnt.c.	163
D.12	Vsebina datoteke msg.h.	163
D.13	UKAZNA LUPINA - Gradnja odjemalca za testno aplikacijo.	164
E.1	Primer opisa značke v VHDL.	167
E.2	Primer opisa protokolnega analizatorja v programskem jeziku Java.	169
E.3	Vmesnik čitalca.	170
E.4	Izvedba vmesnik čitalca z adapterjem.	171
G.1	Vmesnik čitalca.	176

Literatura

- [1] T. Karygiannis, B. Eydt, G. Barber, L. Bunn, in T. Phillips, “NIST Special Publication 800-98: Guidelines for Securing Radio Frequency Identification (RFID) Systems,” NIST, tehnično poročilo, 2007.
- [2] J. Banks, D. Hanny, M. A. Pachano, in L. G. Thompson, *RFID Applied*. Wiley, 2007.
- [3] B. Schneier, *Beyond Fear: Thinking Sensibly About Security in an Uncertain World*. Springer, 2003.
- [4] T. M. Chen in S. Abu-Nimeh, “Lessons from Stuxnet,” *Computer*, zv. 44, št. 4, str. 91–93, apr. 2011.
- [5] European Commission, “RFID Adoption and Implications,” European Commission, DG Enterprise & Industry, tehnično poročilo, 2008.
- [6] J. Landt, “The history of RFID,” *IEEE Potentials*, zv. 24, št. 4, str. 8–11, okt. 2005.
- [7] M. Rieback, B. Crispo, in A. Tanenbaum, “The Evolution of RFID Security,” *IEEE Pervasive Computing*, zv. 5, št. 1, str. 62–69, jan. 2006.
- [8] H. Stockman, “Communication by Means of Reflected Power,” *Proceedings of the IRE*, zv. 36, št. 10, str. 1196–1204, okt. 1948.
- [9] R. J. Klensch, *Electronic identification system*. United States Patent, dec. 1973.
- [10] P. Biggs, T. Kelly, Y. Lozanova, L. P. Chavolla, J. Ponder, R. Sagalbayeva, S. Skvortsova, C. Sund, in L. Srivastava, “ITU Internet Reports: The Internet of Things,” ITU, tehnično poročilo, 2005.
- [11] ISO/IEC 15961:2004, *Information technology – Radio frequency identification (RFID) for item management – Data protocol: application interface*, 2004.

- [12] ISO/IEC 15962:2004, *Information technology – Radio frequency identification (RFID) for item management – Data protocol: data encoding rules and logical memory functions*, 2004.
- [13] ISO/IEC 18000-1:2008, *Information technology – Radio frequency identification for item management – Part 1: Reference architecture and definition of parameters to be standardized*, 2008.
- [14] EPCglobal, *EPCglobal Architecture Framework 1.3*, 2009.
- [15] ISO 9000:2005, *Quality management systems – Fundamentals and vocabulary*, 2005.
- [16] ISO/IEC 9126-1:2001, *Software engineering – Product quality – Part 1: Quality model*, 2001.
- [17] B. Schneier, “The psychology of security,” *Communications of the ACM*, zv. 50, št. 5, str. 128, maj 2007.
- [18] ISO/IEC 27000:2009, *Information technology – Security techniques – Information security management systems – Overview and vocabulary*, 2009.
- [19] R. Anderson, “Why cryptosystems fail,” v *Proceedings of the 1st ACM conference on Computer and communications security CCS '93*. New York: ACM, dec. 1993, str. 215–227.
- [20] B. Schneier, *Secrets and Lies: Digital Security in a Networked World*. John Wiley & Sons, 2004.
- [21] M. Meriac in H. Plotz, “Analyzing a Modern Cryptographic RFID System,” 2010.
- [22] F. Thornton in C. Lanthem, *RFID Security*. Syngress, 2005.
- [23] ODIN Technologies, “RFID Tag Pricing Guide 2010,” ODIN Technologies, tehnično poročilo, 2010.
- [24] D. Trček in P. Jäppinen, *Non-deterministic Lightweight Protocols for Security and Privacy in RFID Environments*. Auerbach Publication, 2009.
- [25] A. Juels, “RFID Security and Privacy: A Research Survey,” *IEEE Journal on Selected Areas in Communications*, zv. 24, št. 2, str. 381–394, 2006.
- [26] —, “The Vision of Secure RFID,” *Proceedings of the IEEE*, zv. 95, št. 8, str. 1507–1508, avg. 2007.
- [27] M. Langheinrich, “A Survey of RFID Privacy Approaches,” *Personal and Ubiquitous Computing*, zv. 13, št. 6, str. 413–421, 2008.
- [28] G. P. Hancke in M. G. Kuhn, “An RFID Distance Bounding Protocol,” v *SECURECOMM '05: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*. Washington: IEEE Computer Society, 2005, str. 67–73.

- [29] D. Trček, "Security and Privacy in RFID Based Wireless Networks," v *Handbook of research on wireless security*, Y. Zhang, J. Zheng, in M. Ma, Ured. Idea Group Inc (IGI), 2008, pogl. XLV, str. 723–731.
- [30] ISO/IEC 27005:2008, *Information technology – Security techniques – Information security risk management*, 2008.
- [31] ISO/IEC 27003:2010, *Information technology – Security techniques – Information security management system implementation guidance*, 2010.
- [32] ISO/IEC 27001:2005, *Information technology – Security techniques – Information security management systems – Requirements*, 2005.
- [33] ISO/IEC 27002:2005, *Information technology – Security techniques – Code of practice for information security management*, 2005.
- [34] M. Bishop, "What is computer security?" *IEEE Security & Privacy Magazine*, zv. 1, št. 1, str. 67–69, jan. 2003.
- [35] R. Barr, "An efficient, unifying approach to simulation using virtual machines," PhD, Cornell University, jan. 2004.
- [36] M. Wille, "Evaluation And Optimization of RFID Transmission Control Strategies," MSc, Swiss Federal Institute of Technology Zürich, 2005.
- [37] C. Floerkemeier in S. Sarma, "RFIDSim – A Physical and Logical Layer Simulation Engine for Passive RFID," *IEEE Transactions on Automation Science and Engineering*, zv. 6, št. 1, str. 33–43, jan. 2009.
- [38] H. Thompson, "Why security testing is hard," *IEEE Security & Privacy Magazine*, zv. 1, št. 4, str. 83–86, jul. 2003.
- [39] J. H. Allen, S. Barnum, R. J. Ellison, G. McGraw, in N. R. Mead, *Software Security Engineering: A Guide for Project Managers*. Addison-Wesley Professional, 2008.
- [40] K. van Wyk in G. McGraw, "Bridging the Gap between Software Development and Information Security," *IEEE Security and Privacy Magazine*, zv. 3, št. 5, str. 75–79, sept. 2005.
- [41] B. Potter in G. McGraw, "Software security testing," *IEEE Security & Privacy Magazine*, zv. 2, št. 5, str. 81–85, sept. 2004.
- [42] S. Ortiz, "How Secure is RFID?" *Computer*, zv. 39, št. 7, str. 17–19, jul. 2006.
- [43] P. Bulić, *Načrtovanje in sinteza digitalnih in mikroprocesorskih sistemov v FPGA*, 2007.
- [44] R. Wilson, "Verifying FPGA designs: Simulate, emulate, or hope for the best?" *EDN*, št. 2, 2009.

- [45] J. J. Labrosse, J. Ganssle, R. Oshana, C. Walls, K. E. Curtis, J. Andrews, D. J. Katz, R. Gentile, K. Hyder, in B. Perrin, *Embedded Software: Know It All (Newnes Know It All)*. Newnes, 2007.
- [46] P. Dillien, “Electrically reconfigurable arrays-ERAs,” v *New Directions in VLSI Design, IEE Colloquium on*, 1989, str. 6.
- [47] R. Thurlow, “RPC: Remote Procedure Call Protocol Specification Version 2,” *RFC 5531*, 2009.
- [48] R. Srinivasan, “XDR: External Data Representation Standard,” *RFC 1832*, 1995.
- [49] —, “Binding Protocols for ONC RPC Version 2,” *RFC 1833*, 1995.
- [50] M. Utting, A. Pretschner, in B. Legeard, “A taxonomy of model-based testing approaches,” *Software Testing, Verification and Reliability*, apr. 2011.
- [51] ISO/IEC TR 15443-2:2005, *Information technology – Security techniques – A framework for IT security assurance – Part 2: Assurance methods*, 2005.
- [52] ISO/IEC 15408-1:2009, *Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model*, 2009.
- [53] J. W. Forrester, *Principles of Systems*, 2nd izdaja. Pegasus Communications, 1968.
- [54] A. S. Tanenbaum in M. van Steen, *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.
- [55] G. P. Hancke in M. G. Kuhn, “Attacks on time-of-flight distance bounding channels,” v *WiSec '08: Proceedings of the first ACM conference on Wireless network security*. Alexandria: ACM, 2008, str. 194–202.

- [56] IDTechEx, “RFID Forecasts, Players and Opportunities 2011-2021,” 2010. Dostopno na: http://www.idtechex.com/research/reports/rfid_forecasts_players_and_opportunities_2011_2021_000250.asp
- [57] ABIResearch, “Total RFID Revenue to Exceed \$5.6 Billion in 2009,” 2009. Dostopno na: <http://www.abiresearch.com/press/1395>
- [58] J. Lettice, “Elvis has left the border: ePassport faking guide unleashed,” *The Register*, 2008. Dostopno na: http://www.theregister.co.uk/2008/09/30/epassport_hack_description/
- [59] B. Schneier, “Cloning RFID Chips Made by HID,” 2007. Dostopno na: http://www.schneier.com/blog/archives/2007/02/cloning_rfid_ch_1.html
- [60] Z. Franken, “Access Control Devices,” 2007. Dostopno na: http://layerone.info/?page_id=9
- [61] —, “Biometric and Token-Based Access Control Systems,” 2008. Dostopno na: <https://www.blackhat.com/presentations/bh-dc-08/Franken/Presentation/bh-dc-08-franken.pdf>
- [62] J. Mandel, A. Roach, in K. Winstein, “MIT Proximity Card Vulnerabilities,” 2004. Dostopno na: http://josephhall.org/tmp/mit_prox_vulns.pdf
- [63] A. Newitz, “The RFID Hacking Underground,” *Wired*, 2006. Dostopno na: <http://www.wired.com/wired/archive/14.05/rfid.html>
- [64] C. Paget, “RSA: Door cards - the enterprise’s weakest link,” 2007. Dostopno na: http://www.infoworld.com/ifwclassic/video/archives/2007/02/rsa_ioactive.html
- [65] “RFID @ wiki.yobi.be.” Dostopno na: <http://wiki.yobi.be/wiki/RFID>
- [66] B. Schneier, “Why Computers are Insecure,” 1999. Dostopno na: <http://www.schneier.com/crypto-gram-9911.html>

- [67] Xilinx, “Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet,” tehnično poročilo, 2007.
- [68] —, “Xilinx.” Dostopno na: <http://www.xilinx.com/>
- [69] Altera, “Altera.” Dostopno na: <http://www.altera.com/>
- [70] Cadence, “Cadence.” Dostopno na: <http://www.cadence.com>
- [71] Eve, “Eve.” Dostopno na: <http://www.eve-team.com/>
- [72] GateRocket, “GateRocket.” Dostopno na: <http://www.gaterocket.com/>
- [73] Mentor Graphics, “Mentor Graphics.” Dostopno na: <http://www.mentor.com/>
- [74] GVHDL, “GVHDL.” Dostopno na: <http://ghdl.free.fr/>
- [75] Xilinx, “ChipScope Pro and the Serial I/O Toolkit.” Dostopno na: <http://www.xilinx.com/tools/cspro.htm>
- [76] Altera, “Verification and Board-Level Tools.” Dostopno na: <http://www.altera.com/products/software/quartus-ii/subscription-edition/verification-board-level/swf-ver-bdlevel.html>
- [77] IEEE, *IEEE Standard VHDL Language Reference Manual: IEEE Std 1076-2008*, 2009, zv. 2008, št. January.
- [78] Xilinx, “Xilinx University Program Virtex-II Pro Development System: Hardware Reference Manual,” Xilinx, tehnično poročilo, 2009.
- [79] —, “Xilinx LogiCORE PLB IPIF (v2.02a) Product Specification,” tehnično poročilo, 2005.
- [80] Sun Microsystems, “ONC + Developer’s Guide,” Sun Microsystems, tehnično poročilo, 2002.
- [81] Xilinx, “MicroBlaze Processor Reference Guide: Embedded Development Kit EDK 10.1i,” Xilinx, tehnično poročilo, 2008.
- [82] Petalogix, “PetaLinux User Guide,” 2009. Dostopno na: <http://www.petalogix.com/resources/documentation/petalinux/userguide>
- [83] UCLinux, “uClinux: Embedded Linux/Microcontroller Project,” 2008. Dostopno na: <http://www.uclinux.org/>