UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Peter Gorišek

TESTIRANJE NOVO VPELJANE PROGRAMSKE REŠITVE ZA PODPORO E-PROIZVODNJE

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: prof. dr. Miha Mraz

Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00455/2009
Datum: 24.06.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **PETER GORIŠEK**

Naslov: **TESTIRANJE NOVO VPELJANE PROGRAMSKE REŠITVE ZA PODPORO E-PROIZVODNJE**

TESTING PROCEDURES OF NEW SOFTWARE SOLUTION USED FOR E-MANUFACTURING

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Kandidat naj v svojem delu predstavi praktične postopke pri validaciji novo vpeljane programske rešitve za podporo v e-proizvodnji, ki je bila instalirana v Krki d.d. Pri tem naj kandidat predstavi strukturo programske rešitve, postopke validacije in testiranja ter konkretne dejavnosti, ki jih je opravil v fazi testiranja.

Mentor:

prof. dr. Miha Mraz

Dekan:

prof. dr. Franc Solina

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00455/2009
Datum: 24.06.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **PETER GORIŠEK**

Naslov: **TESTIRANJE NOVO VPELJANE PROGRAMSKE REŠITVE ZA PODPORO E-PROIZVODNJE**

TESTING PROCEDURES OF NEW SOFTWARE SOLUTION USED FOR E-MANUFACTURING

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Kandidat naj v svojem delu predstavi praktične postopke pri validaciji novo vpeljane programske rešitve za podporo v e-proizvodnji, ki je bila instalirana v Krki d.d. Pri tem naj kandidat predstavi strukturo programske rešitve, postopke validacije in testiranja ter konkretne dejavnosti, ki jih je opravil v fazi testiranja.

Mentor:

prof. dr. Miha Mraz

Dekan:

prof. dr. Franc Solina

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani **Peter Gorišek**, z vpisno številko **63030120**, sem avtor diplomskega dela z naslovom: **Testiranje novo vpeljane programske rešitve za podporo e-proizvodnje**.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Mihe Mraza
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 3.7.2009 Podpis avtorja: _____

Zahvala

Zahvaljujem se svojemu mentorju prof. dr. Mihu Mrazu za potrpežljivost, pomoč ter usmerjanje pri izdelavi diplomske naloge. Hvala tudi kolegom iz Krke d.d., še posebej Gregorju, Denisu ter Andreju.

Posebna zahvala gre moji mami, bratu Mateju, dedku ter Tršinarjevim. Hvala za vse. Upam, da se vam kdaj oddolžim.

Kazalo vsebine

Kazalo slik

Kazalo tabel

Razlaga kratic

CFR 21	Code of Federal Regulations Title 21
CPE	centralna procesna enota
DS	dizajn specifikacija (ang. *Design Specification*)
FDA	Food and Drug Administration
FDS	funkcionalna specifikacija (ang. *Functional Design Specification*)
GAMP	Good Automated Manufacturing Practice
GMP	Good Manufacturing Practice
GxP	splošen term za dobro prakso, kjer »x« označuje specifično področje
HPLC	High performance liquid chromatography
HSM	Hierarchical Storage Management
IQ	kvalifikacija inštalacije (ang. *Instalation Qualification*)
ISO	International Organization for Standardization
ISPE	International Society for Pharmaceutical Engineering
ITT	internet in telekomunikacije
JCAMP	Joint Committee on Atomic and Molecular Physical Data
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
LIMS	Laboratory Information Management System
MES	Manufacturing Execution Systems
ODBC	Open Database Connectivity
OQ	kvalifikacija delovanja (ang. *Operational Qualification*)
PDF	Portable Document Format
PQ	kvalifikacija storilnosti (ang. *Performance Qualification*)
RAID	Redundant array of independent disks
RPC	Remote Procedure Call
SCADA	Supervisory Control And Data Acquisition
SDMS	Scientific Data Management System
SOP	splošni operacijski postopek
TXT	tekstovna datoteka
URS	zahteve uporabnika (ang. *User Requirements Specification*)

VMP	validacijski master plan

POVZETEK

V pričujoči diplomski nalogi predstavimo sistem SDMS, ki služi kot podpora e-proizvodnji. Na začetku spoznamo osnovne lastnosti sistema ter strojne zahteve za postavitev sistema. Sledi opis programskih orodij za zajem podatkov v SDMS sistem. Najprej opišemo agente za zajem podatkov ter enostaven zajem podatkov. Sledi opis bolj kompleksnih instrumentalnih agentov. Kot eno ključnih stopenj pri uveljavitvi sistema v proizvodnji, opišemo validacijski postopek, ki se ga izvaja v Krki d.d. in ustreza vsem predpisanim regulativam. V nadaljevanju diplomskega dela opišemo praktična postopka testiranja komponent SDMS sistema. Predstavimo dokumentirano ter nedokumentirano testiranje, nato pa predstavimo rezultate testiranj.

KLJUČNE BESEDE: SDMS, avtomatski zajem podatkov, validacija računalniških sistemov, testiranje programske opreme

ABSTRACT

In this thesis is introduced the SDMS system used as support of e-manufacturing. At the beginning, the main features are presented and hardware requirements for the system layout. In the next chapter the software tools for data capture are described. The agent for data capture and simple data capture is described at first. This is followed by description of more complex instrumental agents. As one of the key steps in the introducing the system in production the validation procedure is described that it is implemented in Krka d.d. and fits all valid regulative. In the continuations of this thesis the practical process of testing the components of SDMS system is described. The documented and undocumented testing is presented as well as the results of testing.

KEYWORDS: SDMS, automatic data capture, validation of computer systems, software testing

1 Uvod

Dandanes se farmacevtska industrija trudi ohraniti kvaliteto produktov, produktivnost, donosnost iz naložb, ter skladnost s standardi, obenem pa dosegati 10-15% rast za delničarje. To dejansko pomeni, da morajo farmacevtska podjetja letno povečati število kandidatov (primerov zdravila) za končni proizvod, ki vstopijo v proces odobritve, skrajšati odzivni čas glede na razmere na trgu, zmanjšati stroške raziskav in razvoja ter proizvodnje. Celotna pot registriranega zdravila, od raziskav do proizvodnje, mora biti racionalizirana. Če pa hočemo racionalizirati dejavnost podjetja mora podjetje preseči prenekatere ovire kot so naprimer globalizacija in prenos znanja znotraj in zunaj podjetja. Med ovire pa prav tako sodi vseprisotna uporaba papirnatih dokumentov. Posebno breme za farmacevtska podjetja pa predstavlja doseganje in upoštevanje standardov (npr. FDA in GAMP). Štirideset odstotkov celotnih stroškov nastane kot posledica delovanja podjetja v skladu s standardi [1].

Enajsto poglavje CFR standarda 21 se pojavlja kot eden od najbolj zahtevnih predpisov za farmacevtska in biotehnična podjetja. Predpisi kot je 21 CFR (poglavje 11), ki zadevajo kreiranje, vzdrževanje, prenose, shrambo ter spremembe elektronskih zapisov, so dodali nove prioritete farmacevtski industriji. Standard 21 CFR (poglavje 11) skupaj z GxP regulativami prestavlja upravljanje z elektronskimi zapisi v regulativne okvire. Vsak sistem, ki tvori elektronske zapise oz. dokumente, ki so zahtevani po katerem od GxP standardov, je treba preučiti in določiti njegovo skladnost s predpisi standarda 21 CFR (poglavje 11). Potencialno je v farmacevtskih in biotehničnih podjetjih takih sistemov lahko več sto. To so lahko analitični inštrumenti in tehnike, Microsoft Office-ovi dokumenti, laboratorijski informacijski sistemi (ang. *laboratory information management systems* - LIMS), sistemi za nadzor in krmiljenje tehnoloških procesov (SCADA) ter sistemi za upravljanje proizvodnje (ang. *manufacturing execution systems* - MES) [2].

Kot rešitev na prejšnje izzive se je pojavila proizvodnja brez uporabe papirja (ang. *paperless electronic manufacturing*) ali e-proizvodnja. E-proizvodnja omogoča hitro izmenjavo informacij med različnimi fazami proizvodnje, razvoja in raziskav znotraj celega podjetja, kar privede do hitrejših in bolj kvalitetnih rešitev. Zmanjša se vrzel med koncentriranimi območji podatkov, nekakšnimi otoki podatkov, ki se nahajajo ločeno znotraj podjetja. Prvi korak pri pripravi okolja za e-proizvodnjo je elektronski zajem podatkov (npr. podatki o seriji, laboratorijski dnevniki itd.).

Naslednji korak je omogočiti interakcijo med temi različnimi tipi dokumentov. Ko to enkrat dosežemo nam e-proizvodnja v trenutku omogoča izdelavo različnih poročil, pogled nad potekom dela ter transparentnost, da opazimo procese, ki predstavljajo ozka grla. Na enostaven način se lahko preko vmesnikov povežemo na nove ali zunanje sisteme. Za skladnost s standardi se elektronski dokumenti za predpisan čas dolgo hranijo v varnem in dostopnem okolju nad katerim se izvaja sledenje (ang. *audit trail*), da se zagotovi integriteta dokumentov.

V Krki d.d. so se odločili, da v procesu vpeljave e-proizvodnje, vpeljejo SDMS sistem. Slednji pokriva področje zajema, obdelave, uporabe ter hrambe dokumentov v elektronski obliki. S sistemom sem se spoznal v času pred testiranji. Preden sem lahko sodeloval pri testiranjih, sem moral spoznati delovanje sistema. Tako sem v tem diplomskem delu v drugem poglavju opisal sistem SDMS ter programska orodja za zajem podatov. Ločeno sem predstavil agente za zajem podatkov, instrumentalne agente ter način zajemanja tiskanih dokumentov. V tretjem poglavju sem opisal postopek validacije ter postopke, ki si v procesu validacije sledijo. Opisal sem tudi dokumente, ki v procesu validacije nastanejo. V četrtem poglavju sem opisal testiranja pri katerih sem sodeloval. Predstavil sem dva načina testiranja, dokumentiranega ter nedokumentiranega. Na koncu sem podal rezultate testiranj, ter komentiral delovanje in funkcionalnost sistema.

2. SDMS okolje za upravljanje z elektronskimi dokumenti

Znanstveni inštrumenti in računalniške simulacije ustvarjajo velike količine podatkov, ki potrebujejo nove metode za njihovo shranjevanje ter organizacijo. Količine podatkov se vsako leto približno podvojijo, ker pa so inštrumenti za zajem izredno natančni, se tudi kvaliteta podatkov močno izboljšuje. Mnogi znanstveniki pogrešajo dneve, ko so lahko vse rezultate z uporabo pisala spravili na en list, analizo podatkov pa opravili npr. kar z ravnilom. Takrat so bile stvari bolj enostavne in znanstveniki so se lahko osredotočili na svoje delo in jim ni bilo treba biti strokovnjak na področju informatike. Največji problem se kaže pri izgradnji vmesnika za upravljanje s podatki. Znanstveniku je namreč potrebno ponovno omogočiti kontrolo nad podatki [3].

V Krki d.d. so za rešitev teh težav in zadostitvi skladnosti s standardi, ki se nanašajo na upravljanje z elektronskimi dokumenti, vpeljali sistem za upravljanje z znanstvenimi dokumenti (ang. *scientific data management system*), ali na kratko SDMS. Izbrali so proizvajalca Waters Inc., ki je specializiran za področje farmacije, biotehnologije ter kemije.

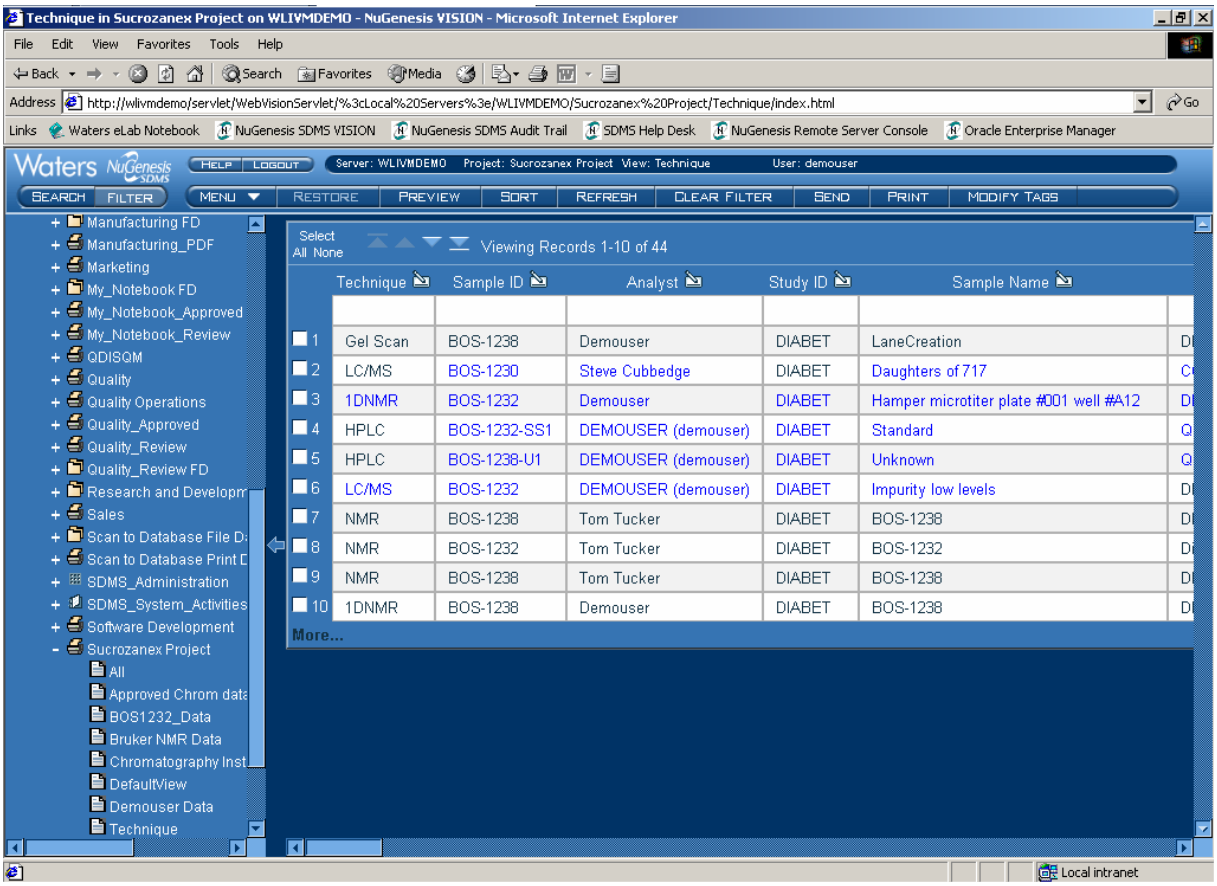
2.1 Waters NuGenesis SDMS

SDMS je sistem za upravljanje z elektronskimi dokumenti. Omogoča upravljanje z njimi skozi njihovo celotno življensko dobo, od njihovega zajema, shranjevanja, iskanja, pa vse do uporabljanja.

NuGenesisov SDMS, ki ga uporabljajo v Krki d.d., je sposoben zajemanja podatkov iz katerekoli laboratorijske naprave ali poslovne aplikacije v trenutku, ko so ustvarjeni. SDMS prav tako lahko zajema vsebino na tiskanih dokumentih iz katerekoli aplikacije na način, s katerim zajame vsebino dokumenta, ne pa samo sliko v smislu fotografije dokumenta. Kje v samem procesu zajemamo podatke, je prikazano na sliki 1.

Slika 1: Lokacije zajema podatkov ter vrstni red obdelave podatkov.

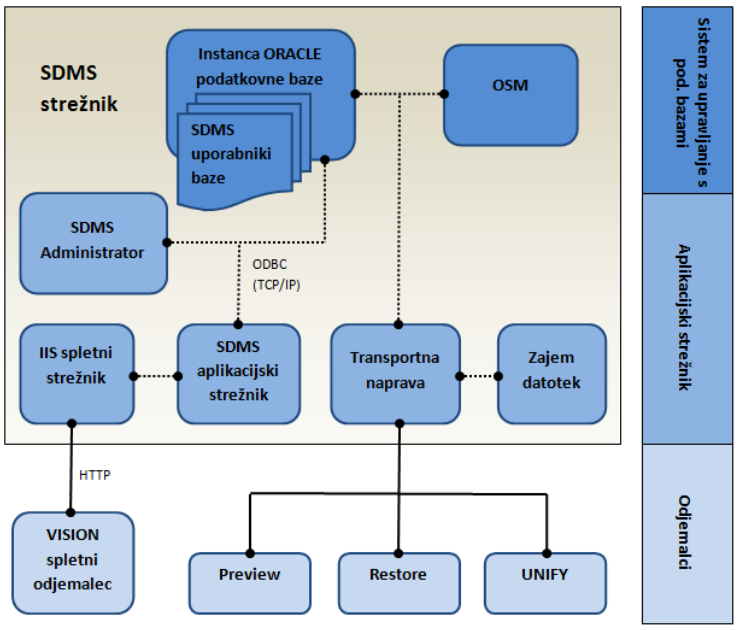
Tipično delo v laboratoriju poteka po naslednjem zaporedju. Na začetku imamo vzorce, o katerih je potrebno zbrati vse informacije. Te vzorce se kasneje analizira in rezultati teh analiz so grobi podatki iz analitičnih inštrumentov. Grobi podatki se obdelajo in na podlagi rezultatov obdelave se sestavi poročilo. Poročila se pregleda, če pa so odobrena, pa se jih tudi podpiše. Vsa dogajanja okoli poročil ter podatkov morajo biti shranjena skupaj na strukturiran način, da se jih lahko kasneje z lahkoto najde ter ponovno uporabi pri nadaljnem dokumentacijskem postopku. Ker so grobi podatki iz inštrumentov ter poročila, ki jih naredijo v laboratoriju, neposredno povezani, se tudi grobi podatki shranijo v SDMS sistem. Ta povezava poročil in grobih podatkov omogoča sklicevanje na te dokumente in njihovo uporabo pri nadaljnih dejavnostih. Pri SDMS sistemu je pomembna tudi dostopnost dokumentov osebam v okviru njihovega razvojnega tima ter glede na pravice tudi ostalim. S tem se olajša izmenjava podatkov ter znanja v podjetju. Kot sem omenil že prej, se podatki zajemajo na dva načina in sicer kot zajem grobih podatkov (datotek) ter zajem podatkov na tiskanih dokumentih. Tiskane dokumente se zajema s pomočjo NuGenesis UNIFY tiskalnika, ki lahko izbrani dokument natisne na dejanskem tiskalniku, ali pa zajeto vsebino dokumenta skupaj z izvlečenimi metapodatki pošlje v SDMS sistem. Grobe podatke se v sistem zajema s pomočjo agentov, ki preverjajo vsebino datotek kamor inštrument shranjuje podatke in glede na to, kako agente nastavimo, zajamejo podatke in iz njih izvlečejo metapodatke ter jih shranijo v SDMS sistem. Variant glede zajema podatkov z agenti je lahko več in jih bom v naslednjih poglavjih tudi podrobneje opisal. Ko so dokumenti enkrat zajeti in shranjeni v SDMS sistemu, si jih je mogoče ogledati s pomočjo spletne aplikacije NuGenesis VISION, katere vmesnik je prikazan na sliki 2 [4].



Slika 2: Vmesnik spletne aplikacije NuGenesis VISION.

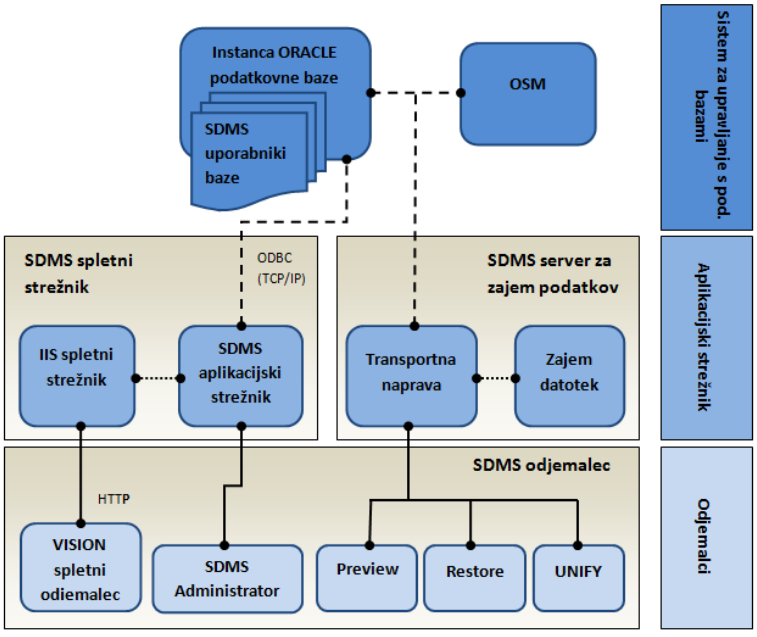
2.2 Arhitektura celotnega sistema

Glede na velikost sistema se lahko prilagodi tudi inštalacijo sistema. Možni sta dve varianti. Prva je, da na enem SDMS strežniku tečejo vsi servisi ter strežniki, ki so potrebni za delovanje sistema. Tak sistem je primeren za manjše skupine ali podjetja in je zmožen podpirati od 5 do 40 uporabnikov oz. inštrumentov. Pri takih sistemih je vedno neznanka količina podatkov, ki bo shranjena v sistemu. Zato morajo biti sistemi zasnovani tako, da so možne povečave prostorske kapacitete. Izhodišča za prvotno oceno pa morajo izhajati na podlagi ocen uporabnikov ter na preteklih izkušnjah. Arhitektura sistema zasnovanega na enem strežniku je predstavljena na sliki 3.



Slika 3: Arhitektura SDMS sistema zasnovana na enem samem strežniku.

Pri drugem načinu instalacije gre za distribuiran sistem strežnikov (glej sliko 4). Vmesna ali aplikacijska plast ter podatkovna plast sta v tem primeru ločeni in delujeta na samostojnih strežnikih. Tako arhitekturo ima tudi SDMS sistem, ki so ga zagnali v Krki d.d., vendar moramo vedeti, da obstajajo tudi druge različice, ki se prilagajajo posameznim zahtevam naročnikov. Za eno stopnjo zmogljivejša arhitektura od arhitekture na enem strežniku je ta, da se hramba podatkov vrši na enem strežniku, aplikacijska plast pa na drugem. Vse variante Watersovih SDMS sistemov pa izhajajo iz teh dveh predstavljenih arhitektur.



Slika 4: Arhitektura večstrežniškega sistema SDMS.

Standardni sistem z distribuiranini strežniki sestoji iz naslednjih komponent, ki lahko delujejo na samostojnih računalnikih:

- **SDMS Oracle podatkovni strežnik**
  - strežnik se uporablja kot centralna hramba podatkov za SDMS sistem,
- **SDMS aplikacijski strežnik**
  - temelji na Javinem aplikacijskem strežniku (J2EE) ter Microsoftovem ISS strežniku,
  - strežnik vsebuje poslovno logiko (transakcije) za VISION klienta,
  - uporablja ODBC vmesnik za komunikacijo z Oracle podatkovno bazo,
- **strežnik za zajem podatkov**
  - na njem tečejo agenti za zajem podatkov,
- **odjemalci**
  - na njih tečejo SDMS Administrator, UNIFY, VISION, Preview in Restore moduli.

2.3 Strojne zahteve

NuGenesisov SDMS sistem zahteva za delovanje pri porazdeljeni strukturi za osnovno delovanje konfiguracijo, ki zadošča kriterijem v tabeli 1. Nekatere od komponent bi lahko delovale tudi pri šibkejših konfiguracijah. Tu še posebno mislimo na naprave na katerih tečejo aplikacije za laboratorijske naprave. Vendar pa bi šibkejša konfiguracija poslabšala odzivnost takih naprav.

Kriteriji za strojno opremo iz zgornje tabele podajajo le minimalne zahteve. Veliko vlogo igra seveda tudi velikost sistema in neposredno tudi število uporabnikov. Za samo hitrost delovanja sistema sta najbolj pomembna hitrost CPE ter velikost spomina. Sistem deluje in je tudi testiran za LAN omrežja, ki uporabljajo TCP/IP v4 protokol. Sama hitrost ter zasedenost omrežja ključno vplivata na hitrost delovanja sistema.

Glede na to, da se pogosto uporabljajo virtualne naprave, kot so naprimer VmWare ter VirtualPC za poganjanje podatkovnih ter aplikacijskih strežnikov (tudi v Krki d.d.), je ponudnik sistema izrecno odsvetoval uporabo le teh, zaradi problemov z zmogljivostjo. Pa tudi sicer sistem kot tak ni validiran za virtualne naprave.

	Oracle strežnik	Spletni aplikacijski strežnik	SDMS Client	Strežnik za zajem podatkov
CPE	Pentium 4 2 GHz	Pentium 4 2 GHz	Pentium 3 500 MHz	Pentium 3 500 MHz
Spomin	2 GB	1 GB	256 MB	512 MB
Operacijski sistem	Windows 2000 Server (SP 4) Windows 2003 Server (SP 1)	Windows 2000 Server (SP 4) Windows 2003 Server (SP 1)	Windows 2000 Server (SP 4) Windows 2003 Server (SP 1) Windows XP	Windows 2000 Server (SP 4) Windows 2003 Server (SP 1) Windows XP
Disk (RAID5 konfiguracija)	100 GB (omogočiti je treba 1GB prostora na vsake 2 GB zajetih grobih podatkov)	10 GB	300 MB	4 GB
Virtualni spomin	2 x velikost spomina	2 x velikost spomina	2 x velikost spomina	2 x velikost spomina

Tabela 1: Strojne zahteve za komponente sistema.

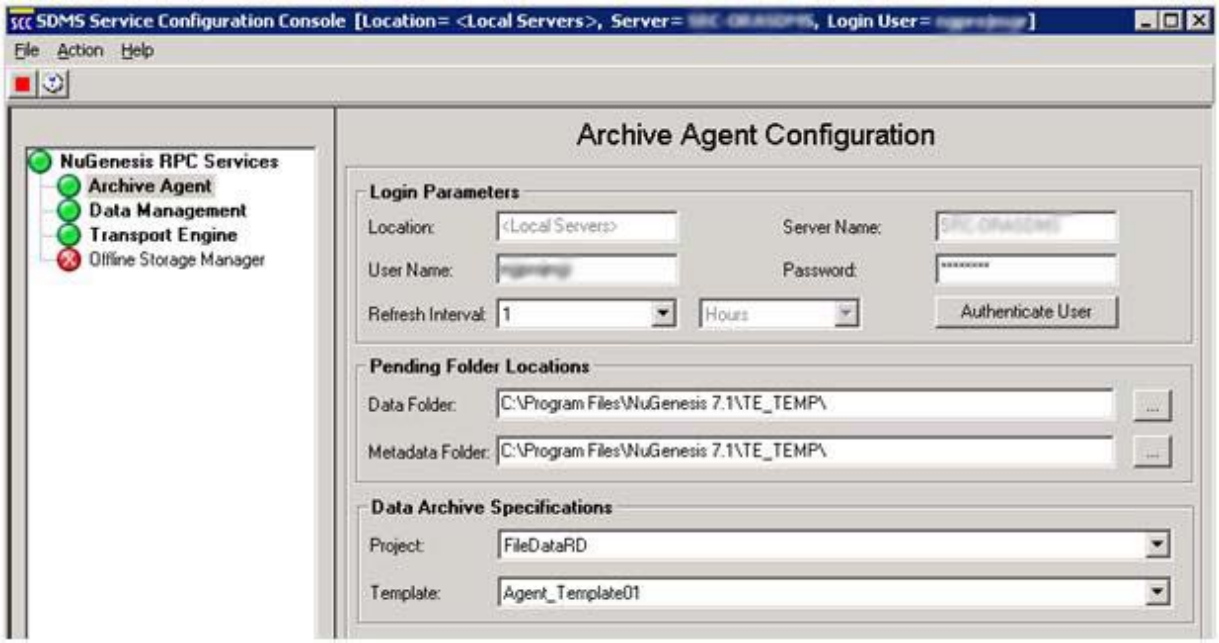
2.4 Agenti za zajem podatkov

Agenti so sami po sebi vtičniki (ang. *plug-ins*), ki neodvisno eden od drugega tečejo pod Nugenesis RPC servisom. Njihova naloga je zajem podatkov in pripadajočih metapodatkov, njihov premik v začasno mapo, nadaljni premik v podatkovno bazo, ter po možnosti shranjevanje v rezervni podatkovni bazi. Agenti so inštalirani kot celota, vendar se da vsakega posebej upravljati in nastavljati. Prav tako se da vsakega izmed njih neodvisno od drugih zagnati ali ustaviti.

Vsak agent sestoji iz razvrščevalnika in izvršilne komponente. Razvrščevalnik daje posle, ki jih mora posamezen agent opraviti, v čakalno vrsto. Izvršilna komponenta agenta jemlje posle iz čakalne vrste, ter jih izvrši. Ker gredo posli vseh agentov v eno čakalno vrsto iz katere jih po tem izvršilne komponente jemljejo (glede na to kateremu agentu je namenjena), je zelo pomembno, da vrsta ni predolga, ker bi se tako zmanjšala odzivnost agentov.

V samem procesu zajemanja datotek sodelujejo štirje agenti:

- **Arhivski agent**
  - v določenih časovnih intervalih pregleduje uporabniško nastavljen direktorij (ponavadi je to mapa v računalniku na katerega je priklopljena laboratorijska naprava) za datotekami, ki so primerne za premik, ter jih premakne v začasno mapo (ang. *pending folder*),
  - ob premiku agent zajame vse mogoče metapodatke in pripravi datoteke transportnemu agentu.
- **Data management agent**
  - ko so podatki uspešno premaknjeni, izbriše podatke iz izvorne lokacije.
- **Transportni agent**
  - v določenih časovnih intervalih pregleduje začasno mapo ter datoteke v njej, ki sta jih tja premaknila arhivski agent, ali pa v primeru zajema podatkov iz tiskanih dokumentov, Unify tiskalnik,
  - agent datoteke skupaj z metapodatki premakne v podatkovno bazo,
  - ko dobi potrditev, da je prenos uspel, zbriše datoteke iz začasne mape.
- **OSM agent** (ang. *offline storage manager*)
  - na določene časovne intervale shrani nove datoteke v dolgoročno podatkovno lokacijo;



Slika 5: »SDMS Service Configuration Console« vmesnik za upravljanje agentov za zajem podatkov.

2.4.1 Arhivski agent

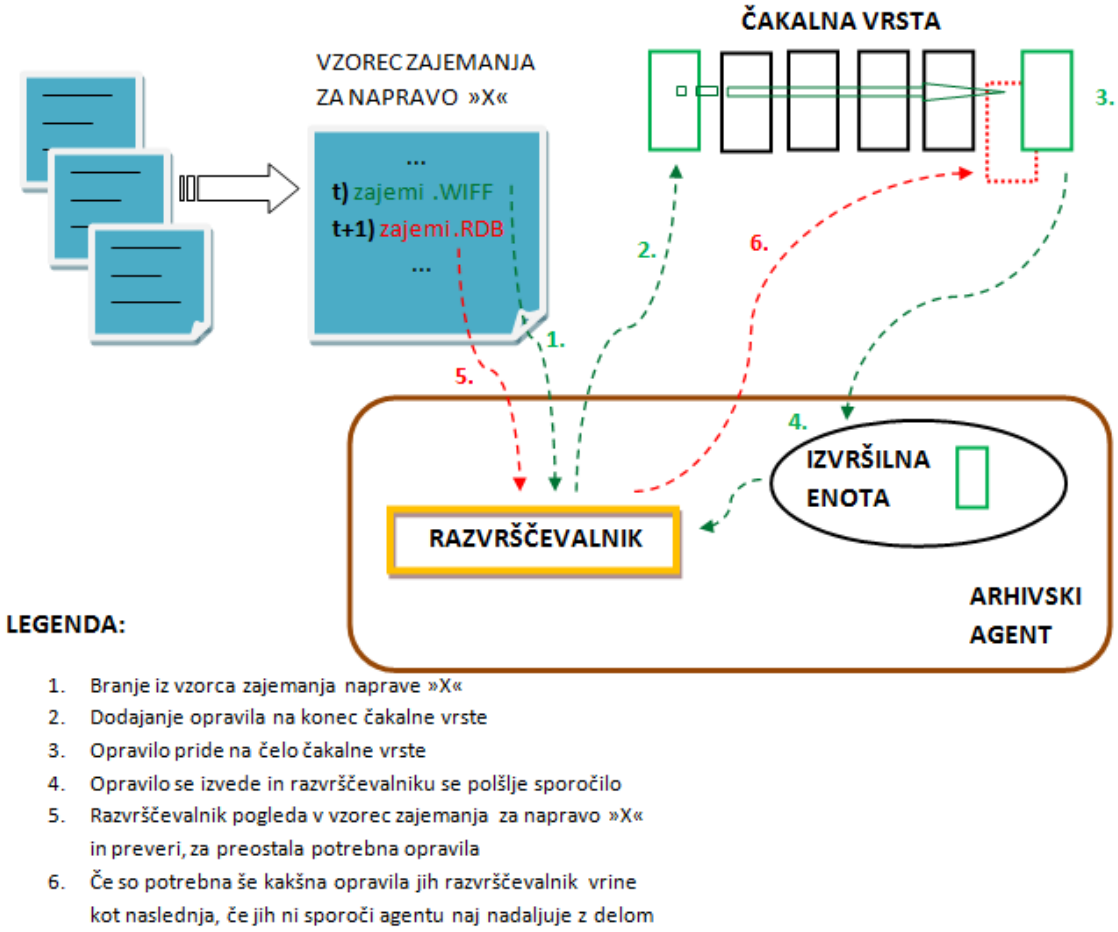
Arhivski agent gleda v prednastavljeno mapo za datotekami, ki so pripravljene za premik v začasno mapo. Ob premiku izlušči metapodatke. V začasni mapi imamo dve ločeni mapi, eno za metapodatke, ter drugo za podatke.

Arhivski agent deluje po navodilih, ki smo jih določili s programom »File capture template builder«, ter jih shranili kot vzorec zajemanja. V teh navodilih je za vsako napravo, iz katere zajemamo podatke, posebej natančno določeno, kako jih bomo zajemali. Ti parametri pa skupaj z številom datotek, ki jih zajemamo, ter strojni konfiguraciji (vključno z hitrostjo omrežja) določajo, kako bodo podatki dejansko zajeti. Čeprav določimo frekvenco proženja zajemanja, je sama hitrost zajemanja odvisna od tega, koliko ostalega dela ima računalnik, na katerem tečejo agenti. Prav zato NuGenesis SDMS ponuja možnost, da imamo lahko več računalnikov, na katerih lahko teče poljubno število agentov, kar pohitri zajemanje.

Arhivski agent torej gleda prvo opravilo v čakalni vrsti in če je namenjeno njemu, ga sprejme ter izvrši, t.j. da pregleda določeno mapo za datotekami, ki so določene v opravilu in so pripravljene za zajem. Agent skopira podatke v začasno mapo, izlušči metapodatke, ter pripravi oboje za shranjevanje v bazo. Ko je opravilo opravljeno, je odstranjeno iz čakalne vrste, razvrščevalniku pa je poslano sporočilo, da je opravilo končano. Agent počaka na odgovor razvrščevalnika. Razvrščevalnik po prejemu sporočila pogleda v vzorec zajemanja za določeno napravo, če so tam še kakšna načrtovana opravila, ki so predvidena, da jih izvede arhivski agent. Če takega opravila ni, pride na čelo čakalne vrste naslednje opravilo, če pa je za dano napravo in agenta v vzorcu zajemanja določeno še kakšno opravilo, ga razvrščevalnik vrine na prvo mesto v čakalni vrsti. Kot rezultat teh neodvisnih, a vendar medsebojno povezanih procesov, se kaže v izvajanju teh procesov, ki niso nujno izvedeni v zaporedju kot, je zapisano v vzorcu zajemanja.

Npr. če so za neko napravo nastavljena vsebinsko in časovno ločena opravila zajemanja datotek npr. tipov ».pdf« ali ».dx«, se recimo najprej izvedejo opravila za zajemanje ».pdf«, ker je njegov naslednji prihodnji čas manjši od drugega. Posli za ».dx« gredo v čakalno vrsto agenta šele potem, ko se posli za ».pdf« zaključijo. Med tem časom pa je velika verjetnost, da v čakalno vrsto agenta vstopijo opravila za drugo napravo. Prav tako ni nujno, da se opravila izvajajo s tako frekvenco, kot je zapisano v opravilu.

S frekvenco, ki jo uporabniško določimo, le nastavimo interval, s katerim bo razvrščevalnik posameznega agenta pošiljal posle v čakalno vrsto. Ko arhivskega agenta zaustavimo, mu razvrščevalnik preneha pošiljati posle, agent pa izvrši še vsa preostala opravila v čakalni vrsti.



Slika 6: Potek izvrševanja poslov arhivskega agenta.

2.4.2 Transportni agent

Transportni agent pregleduje začasno mapo, bolj natančno mapo, kjer so shranjeni metapodatki. V teh datotekah z metapodatki piše, katere datoteke so pripravljene in jih je potrebno prenesti v podatkovno bazo. Ko agent najde tako datoteko z metapodatki, razvrščevalnik v čakalno vrsto pošlje posel v katerem je opisano katere datoteke mora transportni agent premakniti. Agent v bazo premakne metapodatke ter pripadajoče podatke.

Transportni agent deluje neodvisno od vzorca zajemanja. Njegova edina funkcija je vnos metapodatkov ter podatkov v podatkovno bazo in čiščenje začasne mape.

Agenta zaženemo in ustavimo v »Service configuration« konzoli. Perioda osveževanja, ki jo nastavimo v tem vmesniku, predstavlja frekvenco, s katero se agent vrača pregledovati začasno mapo ter poskuša premakniti datoteke, ki jih v prejšnjih poskusih ni uspel.

Transportni agent deluje po enakem principu, kot ostali agenti za zajem podatkov. Sestavljen je iz razvrščevalnika in izvršilne enote. Enako kot pri ostalih agentih, izvršilna enota transportnega agenta izvrši opravilo, ki ji je dodeljeno t.j. da premakne podatke in pripadajoče metapodatke v podatkovno bazo in po uspešnem prenosu izbriše prenešene datoteke iz začasne mape.

Ko transportnega agenta ustavimo, njegov razvrščevalnik takoj preneha z dodajanjem poslov v čakalno vrsto. Agent pa se dejansko ustavi šele, ko izvrši posle, ki so ostali v čakalni vrsti. Če transportnega agenta ustavimo, se podatki ne prenašajo v podatkovno bazo, ampak obtičijo v začasni mapi, vse dokler agenta ponovno ne zaženemo. Ko agenta po zaustavitvi ponovno zaženemo, ta nadaljuje svoje delo po konfiguraciji, ki smo jo nastavili. To neškodljivo zaustavljanje ter zaganjanje agenta pa lahko koristno izrabimo. Z določanjem časovnih intervalov delovanja transportnega agenta, lahko zmanjšamo obremenitev omrežja. Npr. podatke čez dan premikamo samo v začasno mapo, čez noč, ko pa je omrežje manj obremenjeno, pa zaženemo arhivskega agenta, ki premika podatke v bazo.

2.4.3 Agent za upravljanje s podatki

Funkcija agenta za upravljanje s podatki ali »data management« agenta, je brisanje podatkov iz začasne mape. To se zgodi potem, ko so podatki od tam že bili uspešno prenešeni na naslednjo načrtovano lokacijo. Tako kot ostali agenti tudi ta glede na vzorec zajemanja gleda določene mape in išče datoteke, ki so kandidatke za izbris. Ko agent naleti na datoteko, ki je glede na vzorec zajemanja kandidatka za izbris, izvede serijo preverjanj ali je datoteka bila uspešno prenešena na naslednjo lokacijo. Če preverjanja uspejo, agent datoteko izbriše. Če pa katero od preverjanj ne uspe, pa agent te v nobenem primeru ne izbriše, ampak delo nadaljuje po navodilih iz vzorca zajemanja.

Tako kot ostali agenti je tudi »data management« agent sestavljen iz razvrščevalnika ter izvršilne komponente. Razvrščevalnik pregleduje določeno lokacijo (običajno je to začasna mapa) za datotekami, ki so kandidatke za izbris.

Ko tako datoteko najde, razvrščevalnik agenta doda novo opravilo v čakalno vrsto. Ko opravilo pride na začetek čakalne vrste, ga izvršilna komponenta prebere, izvede preverjanja in ob uspešnih preverjanjih na koncu datoteke tudi izbriše. V vsakem takem postopku je določeno največje število datotek, ki se jih obdela in izbriše. Če pri pregledovanju prepoznamo več kandidatov, kot je maksimalno določeno, jih izvršilna enota obdela le toliko, kot je največ določeno. Ostale datoteke pa se obdela v dodatnih ponovitvah postopka, vse dokler niso vse obdelane. V primeru inštalacije v Krki d.d. je ta meja nastavljena na sto datotek in še omogoča spodobno odzivnost sistema.

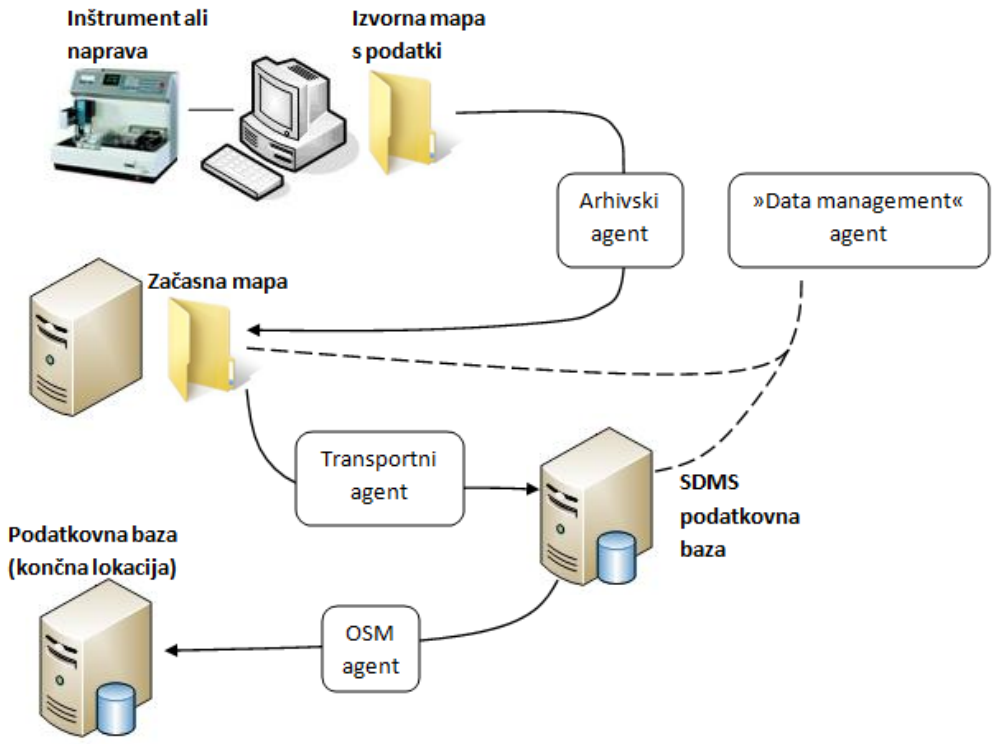
2.4.4 OSM agent

OSM agent je zadolžen za premikanje datotek iz podatkovne baze sistema na dolgoročno lokacijo, kjer bodo podatki shranjeni do njihovega dokončnega izbrisa. Ta agent tako vrši zadnjo v vrsti nalog zajema podatkov v sistem SDMS. Podpira shranjevanje na mnoge podatkovne medije kot so trdi diski, HSM naprave, RAID diskovna polja, CD ali DVD podatkovne knjižnice ter ostale. OSM agent kopira podatke iz SDMS baze na drugo lokacijo, prvotno kopijo podatkov v SDMS bazi pa izbriše. Podatke izbriše podobno kot »data management« agent šele po tem, ko preveri, če so se podatki uspešno prenesli. Istočasno pa posodobi metapodatke, ki ostanejo v SDMS bazi, s podatki o novi lokaciji podatkov. Pomembno je, da OSM agent teče na isti napravi na katero agent premika podatke. Samo izvajanje premikanj je podobno izvajanju operacij ostalih agentov. Razvrščevalnik OSM agenta pregleduje bazo in išče nove datoteke. Ko najde datoteke primerne za premik, doda opravilo v čakalno vrsto, od koder ga razvrščevalnik pobere ter izvrši.

2.4.5 Enostaven zajem podatkov

Očitno je, da imamo pri različnih napravah opravka z različnimi podatki. Npr. če bi podatke zajemali iz tabletirne naprave, ki z močnim pritiskom stiska surovine v končni produkt (tableto), bi bili ti podatki veliko manj kompleksni kot rezultati kromatografije. Bolj ko so kompleksni podatki, več metapodatkov nosijo s sabo ter so običajno sestavljeni iz več datotek, vsaka izmed njih pa hrani specifične podatke. Za primer enostavnega zajema podatkov si

zamislimo tabletirno napravo, ki nase priključenemu računalniku pošlje tekstovno datoteko z vsemi podatki. Arhivski agent bo torej po vzorcu zajemanja iz datoteke izluščil metapodatke in jih skupaj z originalno datoteko premaknil v začasno mapo. Ti podatki so sedaj pripravljeni za vnos v bazo. Tu ni potrebna nobena konverzija tipa datoteke ali kakšno drugo predprocesiranje, ker za vse poskrbi arhivski agent. Na sliki 7 lahko vidimo kako poteka zajem enostavnih podatkov.



Slika 7: Potek enostavnega zajema podatkov ter vloga posameznih agentov.

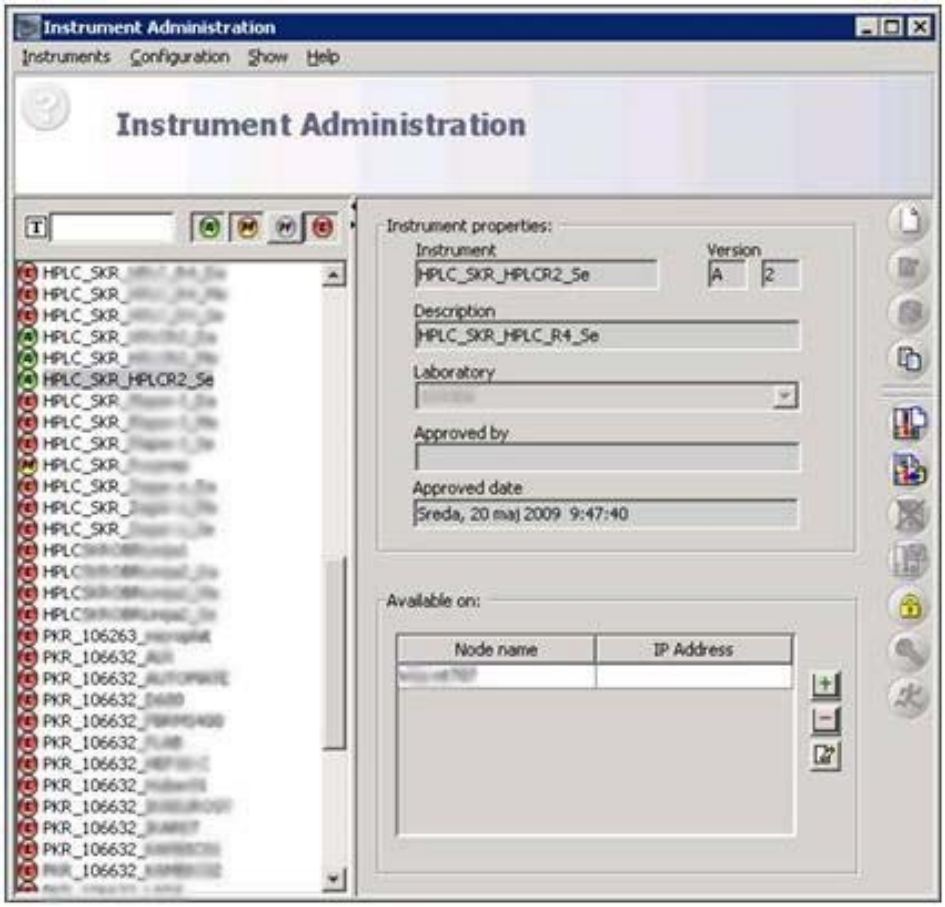
2.5 Instrumentalni agenti

Instumentalni agenti so programska orodja, ki ponujajo napredno strategijo zajemanja bolj kompleksnih podatkov iz različnih virov. Agenti opravljajo naslednje funkcije:

- iskanje podatkov po prednastavljenih lokacijah oz. mapah in sicer po kriteriju, ki ga sami določimo,

- kopiranje ustreznih map iz izvorne lokacije ter procesiranje datotek z izluščanjem metapodatkov in vsebine ter imen map, konverzijo v standarne formate ter drugo predprocesiranje,
- shranjevanje podatkov na začasno lokacijo, ki jo sami nastavimo.

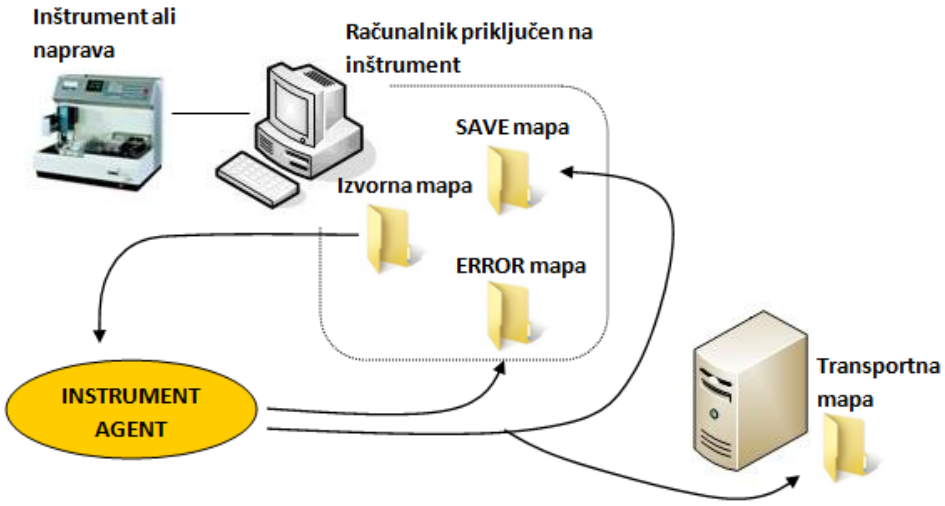
Za vsako napravo iz katere zajemamo podatke kreiramo enega ali več instrumentalnih agentov. Npr. če bi hoteli, da se določeni podatki iz posamezne naprave, zajemajo na svoj način, bi za to ustvarili več agentov. Vsak od teh agentov pa bi določene podatke zajemal glede na to, kako bi ga nastavili. Podobno storimo tudi takrat, ko hočemo, da se zajeti podatki shranjujejo v ločene SDMS projekte. Vsak agent lahko namreč shranjuje podatke le v en projekt. Agente kreiramo in nasploh z njimi upravljamo z aplikacijo »Instrument administration«, katere vmesnik lahko vidimo na sliki 8.



Slika 8: Aplikacija »Instrument Administration«.

2.5.1 Umestitev instrumentalnih agentov

Instrumentalni agent je samostojna nadgradnja arhivskega agenta. Podatke zajema iz računalnika, na katerega je priklopljena določena naprava, jih obdela, ter premakne v transportno mapo. Podatke, ki jih prenaša v transportno mapo, za določen čas shrani v dve mapi na izvornem računalniku. V »save« mapo podatke premakne, če prenos uspe, v nasprotnem primeru pa se podatki shranijo v »error« mapo. Pot podatkov iz transportne mape do končne lokacije poteka tako, kot v primeru zajema enostavnih podatkov. To pomeni, da podatke iz transportne mape zajema arhivski agent.



Slika 9: Umestitev instrumentalnega agenta.

2.5.2 Proženje in način zajemanja podatkov

Vsakega agenta lahko zaženemo in ustavimo ročno, lahko pa jih nastavimo, da se zaženejo avtomatsko takrat, ko se zažene računalnik na katerem agenti tečejo (v aplikaciji »Instrument administration«). Prav tako vsakemu agentu nastavimo interval po katerem pregleduje izvorno mapo s podatki. Dejanski proces obdelave ter premika datotek se proži glede na dejavnike znotraj map, ki jih pregledujemo. Lahko zajemamo cele mape, ali pa posamezne datoteke.

2.5.2.1 Zajemanje map

Pri zajemanju podatkov lahko instrumentalnega agenta nastavimo tako, da podatke zaradi zmanjšanja nevarnosti izgube le teh, shrani v »save« ali »error« mapo. V »save« mapo jih shrani, če prenos uspe ter v »error« mapo, če prenos ne uspe. Namreč, ko agent enkrat zajame podatke, jih iz izvirne lokacije izbriše, če pa se pri prenosu v transportno mapo pojavijo kakšne težave in prenos ni mogoč, se ti podatki izgubijo. Zato se zaradi varnosti uporabljata ti dve mapi, za kateri nastavimo tudi čas, po katerem se te dve mapi počistita. Ko zajemamo celotne mape, moramo paziti, da pri premikanju v transportno mapo tam že ne obstajajo datoteke z enakimi imeni, ker v tem primeru agent podatkov ne more premakniti. Zato moramo paziti pri časovnih nastavitvah ter omogočiti dovolj časa arhivskemu agentu, da premakne podatke iz transportne mape, preden vanjo pošiljamo nove. Podoben problem se pojavi, ko zajemamo velike mape, v katerih se lahko ime kakšne datoteke ponovi. Tudi v tem primeru agent ne more zajeti podatkov. Najboljša rešitev obeh težav je stiskanje zajetih podatkov v eno ZIP datoteko, kar lahko stori instrumentalni agent sam. Tako agent v bistvu premika le eno datoteko, kar zelo zmanjša ali celo odpravi verjetnost pojavitve enakih imen datotek ter ohrani drevesno strukturo zajetih podatkov.

Instrumentalni agenti omogočajo naslednje možnosti zajema map:

- **Zajem celotnih map:**
  Pri tej možnosti zajamemo celotno mapo ter vse njene podmape. Zajem podatkov lahko sprožimo glede na stanje mape v izvorni mapi ali podmapah, ob kreiranju datoteke z določeno končnico, ali pa ob pojavitvi določenega števila datotek.
- **Zajem od nivoja navzdol:**
  Tu zajamemo mapo in njene podmape glede na pojavitev datoteke z določenim delom imena. Določimo lahko vse od celotnega imena datoteke do njene končnice. To datoteko imenujemo sprožitvena datoteka. Ko je taka datoteka najdena, se zajame mapa, v kateri je ta sprožitvena datoteka ter njene podmape. Ali drugače povedano prvi nivo zajema je mapa s sprožitveno datoteko.

- **Zajem spremenjenih map:**
  Zajamemo samo en nivo mape glede na to, kdaj je bila mapa nazadnje spremenjena. Podmap te mape ne zajemamo, vsebina pa za razliko od zgornjih dveh možnosti ostane v mapi, torej se ne izbriše. Vsebina je prvič zajeta, ko zaženemo instrumentalnega agenta ter vsakič naslednjič, ko mapa ustreza kriteriju zadnje spremembe.

2.5.2.2 Zajem datotek

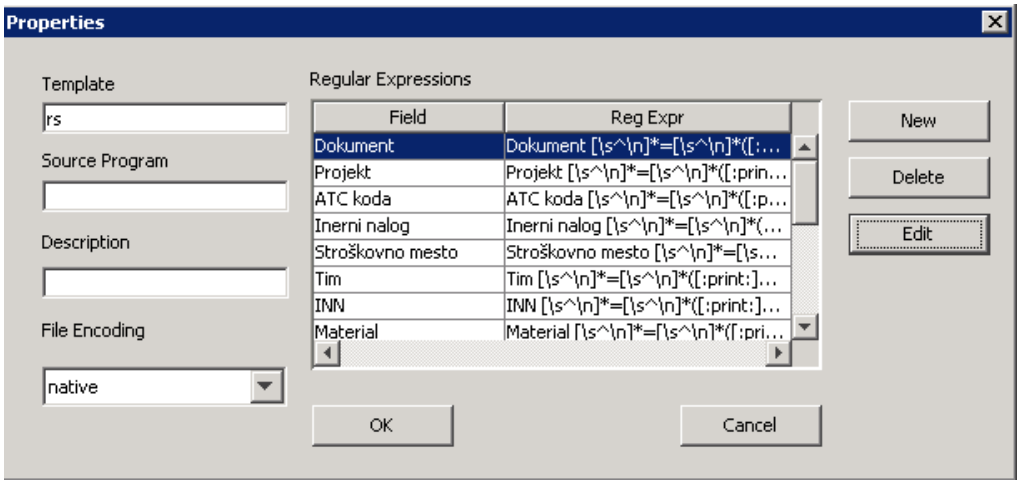
Posamezne datoteke lahko zajemamo na naslednje načine:

- **Zajem datotek v mapi:**
  Pri tem načinu zajamemo vse datoteke v določeni mapi, ali pa le datoteke z določeno končnico oz. datoteke določenega tipa. Določimo lahko koliko časa mora preteči od najdbe datoteke do takrat, ko jo instrumentalni agent zajame. Ta parameter je dobro nastaviti na tako vrednost, da preprečimo zajemaje datoteke v času, ko se še zapisuje.
- **Zajem določenih tipov datotek v mapi:**
  Na ta način lahko določimo več tipov datotek, ki jih bomo zajemali. Datoteke se obdelajo po alfanumeričnem vrstnem redu, ne pa glede na tip datoteke.
- **Zajem spremenjenih datotek:**
  Podobno, kot pri zajemu map, zajemamo datoteke glede na to, kdaj so bile nazadnje spremenjene ali kdaj so bile ustvarjene.

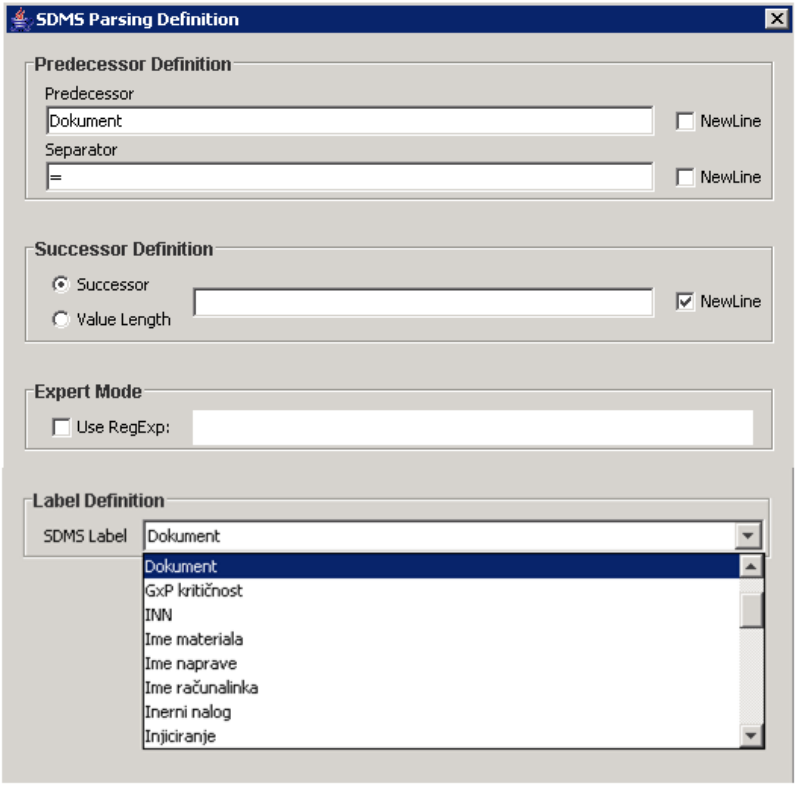
2.5.3 Luščenje metapodatkov

Metapodatki so opisne informacije o podatkih, ki opisujejo izmerjene atribute, njihova imena, enote, natančnost, metode in še mnogo več. Lahko bi jim rekli kar podatki o podatkih. Najbolj pomembni so metapodatki, ki opisujejo kako so bili podatki izmerjeni, zajeti ter izračunani. Metapodatki so ključnega pomena pri urejanju in iskanju po zbirkah podatkov. Če podatke analiziramo z generičnimi orodji, morajo taka orodja razumeti podatke. Nekdo ne more nekemu orodju predstaviti niz bitov ter pričakovati, da bo to orodje intuitivno prepoznalo kje in kakšni so rezultati ter kaj pomenijo.

Orodja za obdelavo podatkov morajo poznati metapodatke, šele takrat lahko izvedejo operacije nad njimi. Npr., če rečemo samo, da imamo datoteko, o tej datoteki ne moremo povedati veliko. Če pa povemo, da je to JPEG datoteka, vemo, da je to bitno polje v standardiziranem formatu. JPEG datoteka se začne z glavo, ki opisuje vrstni red podatkov. Pogostokrat opisuje kamero, časovne parametre ter program s katerim je bila slika kreirana [5].



Slika 10: Vmesnik za določanje pozicije metapodatkov.



Slika 11: Dodeljevanje vrednosti metapodatkov SDMS labelam.

Podobno velja za vse vrste podatkov, zato v sistemu SDMS izluščamo metapodatke ter jih shranjujemo ločeno poleg originalnih datotek. Zelo nazorno so metapodatki prikazani v spletni aplikaciji Vision (slika 2). V isti aplikaciji lahko nastavimo različne filtre, kjer glede na vrednosti metapodatkov določimo kateri dokumenti bodo prikazani. Na slikah 10 in 11 vidimo vmesnika iz programa »Instrument Administration«, kjer določimo, kje v datoteki so metapodatki, ter kako jih naj zajame.

V SDMS sistemu poznamo izluščanje metapodatkov iz imena datoteke ter iz vsebine datoteke.

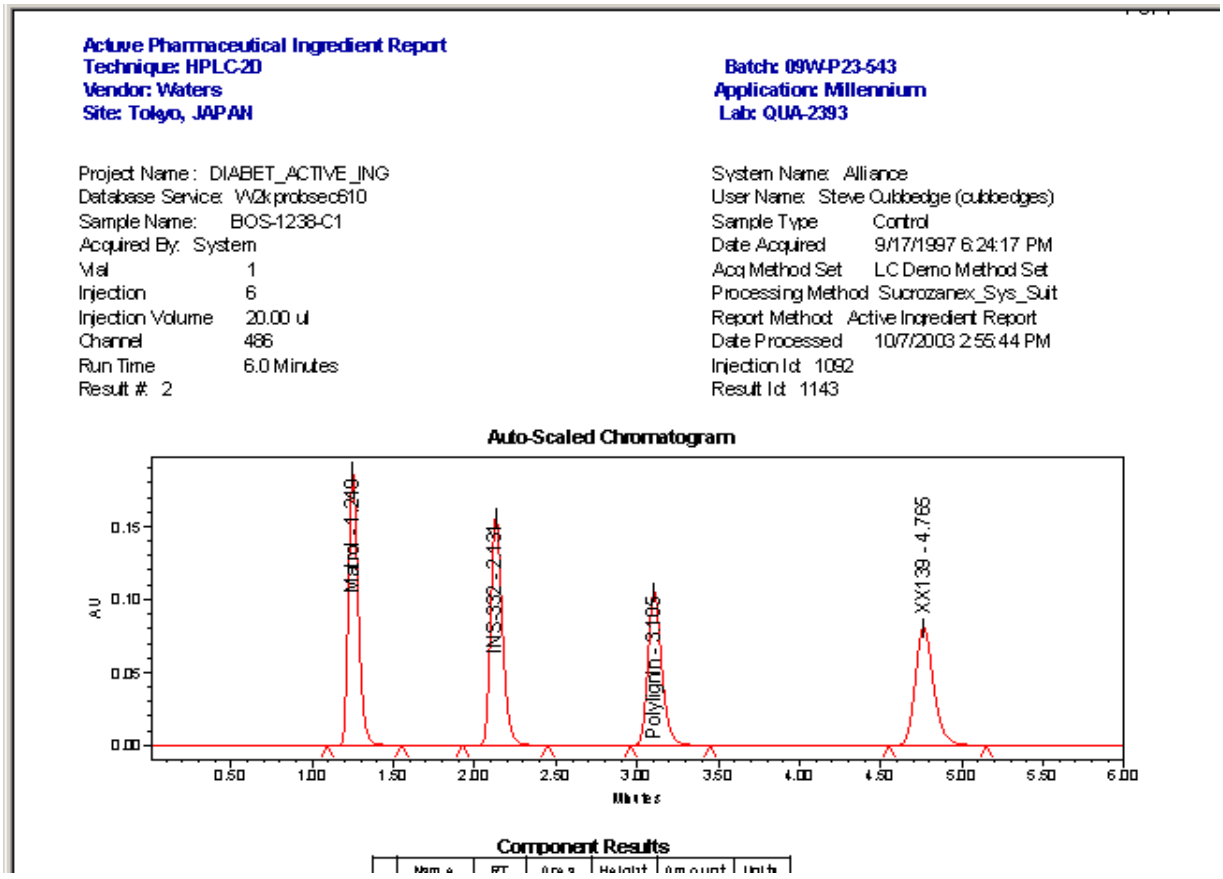
- **Izluščanje metapodatkov iz imena datoteke**
  Že celotna pot do datoteke ter njeno ime, nosita pomembne podatke o sami vsebini datoteke. Pomembnosti teh podatkov ne gre zanemarjati, zato jih prav tako zajamemo in iz njih izluščimo metapodatke.
  Primer: Iz imena datoteke » **123456_MEDIKAL_20_MG_J43450_1_ZOČ.pdf**« lahko določimo naslednje metapodatke:
  - **123456**: številka stroškovnega mesta,
  - **MEDIKAL_20_MG**: ime proizvoda,
  - **J43450:** številka serije,
  - **1:** notranja klasifikacija dokumenta,
  - **ZOČ:** vrsta dokumenta, v tem primeru je to zapisnik o čiščenju.

  Ko v nastavitvah določimo, kje v imenu se posamezen podatek nahaja, ga izluščimo ter določimo kateri vnaprej določeni labeli v SDMS sistemu pripada.

- **Izluščanje metapodatkov iz vsebine datoteke**
  Iz same vsebine lahko izluščimo največ metapodatkov. Lokacijo metapodatkov v vmesniku določimo s standardnimi izrazi. Z njimi določimo kateremu predhodniku sledi podatek, ki ga iščemo. V SDMS vmesniku »parsing definition« določimo kateri labeli podatki pripadajo.

  Primer: Na sliki 12 imamo prikazan začetek poročila, ki ga je izdelal kromatograf. Kot vidimo, so na začetku navedeni metapodatki, ki opisujejo podatke na dokumentu, nato pa rezultati kromatografije v obliki grafa. Instrumentalni agent pregleda tak dokument in če smo določili, da predhodniku »Project Name« in ločilu »:« sledi metapodatek, ki

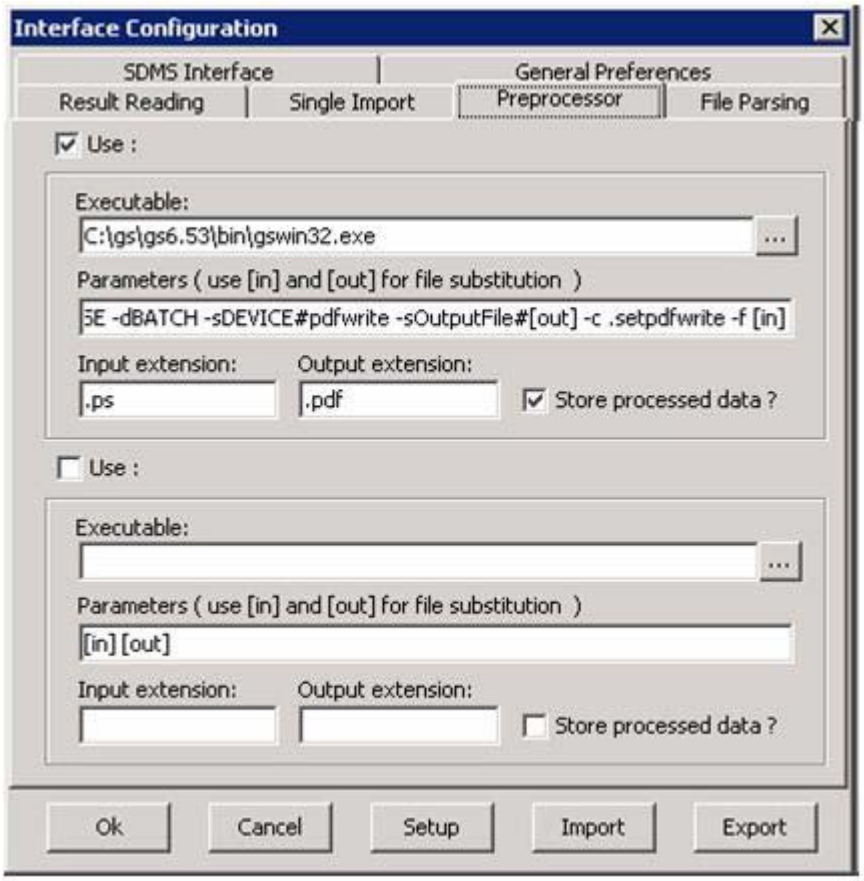
opisuje projekt kateremu dokument pripada, se v predhodno nastavljeno labelo v SDMS sistemu »Projekt« shrani vrednost »DIABET_ACTIVE_ING«.



Slika 12: Primer poročila, ki ga je izdelala laboratorijska naprava.

2.5.4 Predprocesiranje in konverzija

Pri možnosti predprocesiranja lahko določimo ali bo instrumentalni agent pred zajemom pognal največ dva zunanja programa, ki bosta obdelala podatke. Določimo lahko le dva programa, ker bi večje število takih programov močno upočasnilo zajemanje podatkov. Npr. tu lahko poženemo program, ki bo iz datoteke prebral metapodatke ter jih shranil v posebno datoteko. Tako potem agentu ni potrebno izluščati metapodatkov. Vendar pa je to samo ena od možnosti, ki jo lahko izkoristimo.



Slika 13: Primer nastavitve predprocesiranja datoteke.

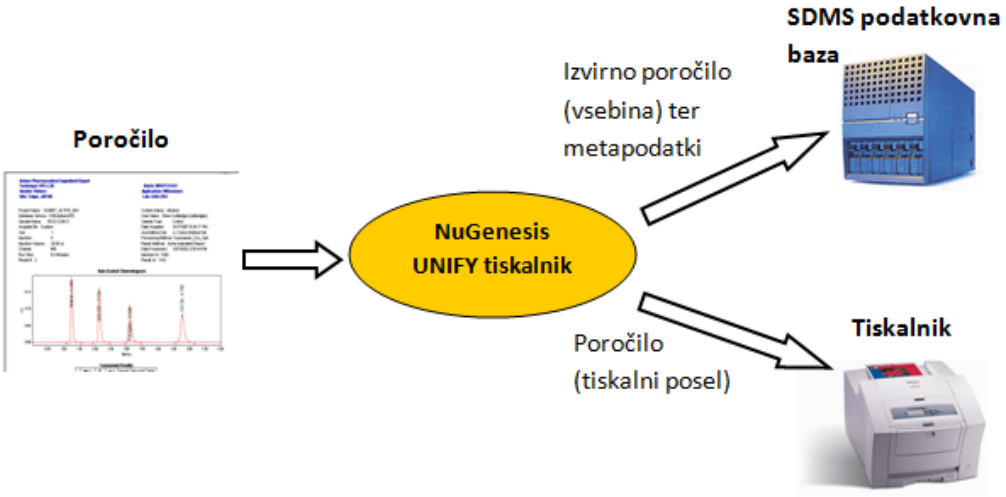
Če hočemo predprocesirati izbrano datoteko, moramo napovedati s katerim programom jo bomo obdelali, ter poznati ukaze ukazne vrstice za izbrani program, da lahko določimo parametre delovanja programa. Na sliki 13 lahko vidimo primer, ko agent pokliče program GhostScript, ki iz datoteke vrste Adobe PostScript® (.ps) kreira PDF datoteko.

Ukazi in parametri za delovanje programa so v drugem vpisnem oknu od zgoraj, pod napisom »Parameters«.

Še ena od možnosti, ki jih ponujajo instrumentalni agenti, je konverzija tipov datotek. Agenti podpirajo večino tipov datotek, ki se jih uporablja pri raziskovalni dejavnosti ter vse pomembnejše komercialne tipe. Največkrat pa se uporablja konverzija tipov v univerzalen standard JCAMP, ki podpira shranjevanje kemijskih ter spektroskopskih podatkov.

2.6 Zajem vsebine na tiskanih dokumentih

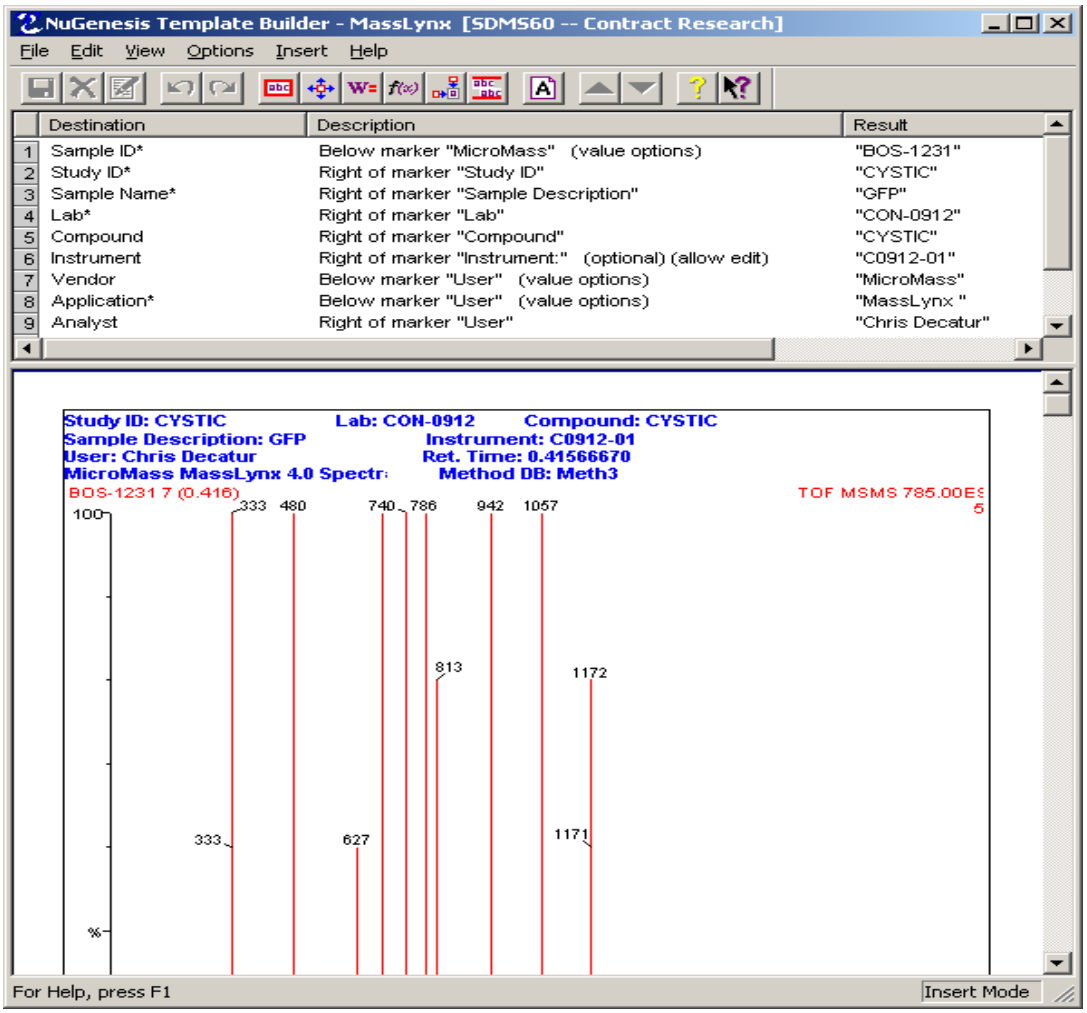
Pri zajemanju vsebine na tiskanih dokumentih v SDMS sistem zajemamo podatke s pomočjo NuGenesis UNIFY tiskalniškega gonilnika. Podatke zajemamo iz katerekoli vrste tiskanih dokumentov, npr. Microsoft Office-ovih dokumentov, PDF dokumentov ter TXT datotek. »Unify« gonilnik lahko pošlje tiskalni posel na dejanski tiskalnik t.j. na privzeti tiskalnik v operacijskem sistemu, ali pa podatke obdela ter jih pripravi za vnos v SDMS podatkovno bazo.



Slika 14: Umestitev NuGenesis UNIFY tiskalnika.

Možnosti zajema podatkov v sistem SDMS je pri tem načinu veliko, zato bom predstavil le najbolj pogosto uporabljenega. Največkrat se Unify gonilnik uporablja tako, da iz tiskanih dokumentov na podlagi pravil, ki jih lahko shranimo kot vzorec (slika 15), izlušči metapodatke.

V SDMS sistem, bolj natančno v mapo, iz katere v sistem transportni agent premika datoteke, shranjuje posebne datoteke, ki so sestavljene iz originalne vsebine ter izluščenih metapodatkov. Lahko bi rekli, da shranjujemo datoteko v kateri sta skriti dve datoteki, ena z metapodatki ter druga z originalnim dokumentom. Seveda obstaja tudi možnost, da originalni dokument konvertiramo v datoteko vrste PDF ter jo skupaj z metapodatki shranimo v sistem.



Slika 15: Aplikacija za določanje vzorca lokacij metapodatkov.

3. Validacija sistema

Po spoznavanju SDMS okolja sem sodeloval pri testiranju kvalifikacije ustreznosti delovanja sistema. Preden sem lahko sodeloval pri validiranju sistema, sem bil jasno seznanjen in poučen o tem, kako se validacija izvaja ter zakaj je potrebna. Namreč samo testiranje delovanje sistema kot del validacije ni tako zahtevno kot same priprave na testiranje. Tu bi izpostavil ogromno dokumentacije, ki pri postopku validacije nastane, ter same priprave na izvajanje testiranj. Ne smemo pa pozabiti, da mora biti cel postopek validacije računalniškega sistema izpeljan v skadu s predpisanimi standardi. Tudi izbrani ponudnik sistema mora delovati v skladu s standardi in predpisi, ki veljajo v farmacevtski industriji.

3.1 Validacijski postopek

Cilj postopka validacije delovanja računalniškega sistema je dokumentirano dokazati, da je izgradnja sistema potekala v skladu z zahtevami, podanimi v tem dokumentu, da je predaja oziroma prevzem sistema ustrezen, ter da sistem v eksploatacijski dobi uspešno obvladujemo. Prav tako je potrebno pokazati, kako bomo potrebne podatke po izteku življenjske dobe sistema uspešno prenesli na nov sistem, oziroma jih obvladali tudi po ukinitvi sistema.

Namen validacijskega postopka je podati navodila za izvajanje procesa validacije za računalniške sisteme. Validacijske aktivnosti potekajo prek celotne faze izgradnje projekta in morajo ustrezati zahtevam, priporočilom in regulativi na tem področju.

Postopek je namenjen osebam, ki

- so odgovorne za pripravo validacijske dokumentacije,
- so odgovorne za odobritev validacijske dokumentacije,
- sodelujejo pri validacijskem procesu,
- kontrolirajo validacijski proces,
- so kakorkoli vpleteni v proces kvalifikacije (uporabnikom, razvijalcem…).

3.2 Področje uporabe validacijskega postopka

Validaciji so podvrženi računalniški sistemi, ki imajo posreden ali neposreden vpliv na kvaliteto proizvoda, nadzorujejo ali upravljajo proizvodni sistem. Sem spada tudi SDMS sistem. Postopek se nanaša na nove računalniške sisteme, delno pa po izvedeni oceni kritičnosti pokriva tudi obstoječe stare sisteme. Sistemi, ki jih moramo validirati so:

- laboratorijski sistemi,
- sistemi za vodenje procesov,
- poslovni sistemi z vplivom na kvaliteto proizvoda-procesa,
- računalniško vodeni stroji,
- potrebna infrastruktura, ki zagotavlja nemoteno in pravilno delovanje teh sistemov.

V odvisnosti od ocene kritičnosti sistema je odvisen način pristopa k validaciji sistema in količina izvedenih aktivnosti. Prav tako je v odvisnosti od kategorije sistema odvisen način pristopa k validaciji sistema. Kategorizacija sistema je odvisna od njegove unikatnosti. Bolj ko je sistem unikaten oziroma prilagojen našim potrebam, večja je potreba po validacijskih aktivnostih. Glede na to, da je bil Watersov SDMS sistem zelo prilagojen potrebam združbe Krka d.d. ter sodi med kritične sisteme, je bilo potrebno temeljito izpeljati vse aktivnosti v okviru postopka validacije sistema.

Sistemi, ki so v uporabi že vrsto let, se ne kvalificirajo po klasičnem validacijskem življenjskem ciklu. Zbere in preveri se obstoječa dokumentacija sistema. Pripravijo se specifikacije oziroma opisi sistema ter postopki potrebni za nemoteno delo in vzdržavanje. Na podlagi ocene GMP kritičnosti se pripravijo testni postopki za izvedbo funkcionalnega testiranja sistema. Zaključno poročilo temelji na pregledu sistema skozi njegovo življenjsko dobo in izvedenih testnih procedurah.

Za validacijo posameznih sistemov je v Krki d.d. odgovorna Komisija za validacije, ki postavi prioritete in odobri potrebne resurse za validacijske aktivnosti sistemov. Odgovornosti za pravilno izvedbo posameznih validacijskih aktivnosti so na strani odgovornega vodje projekta, službe za ITT, sektorja za upravljanje kakovosti, ter odgovornega uporabnika kot lastnika sistema.

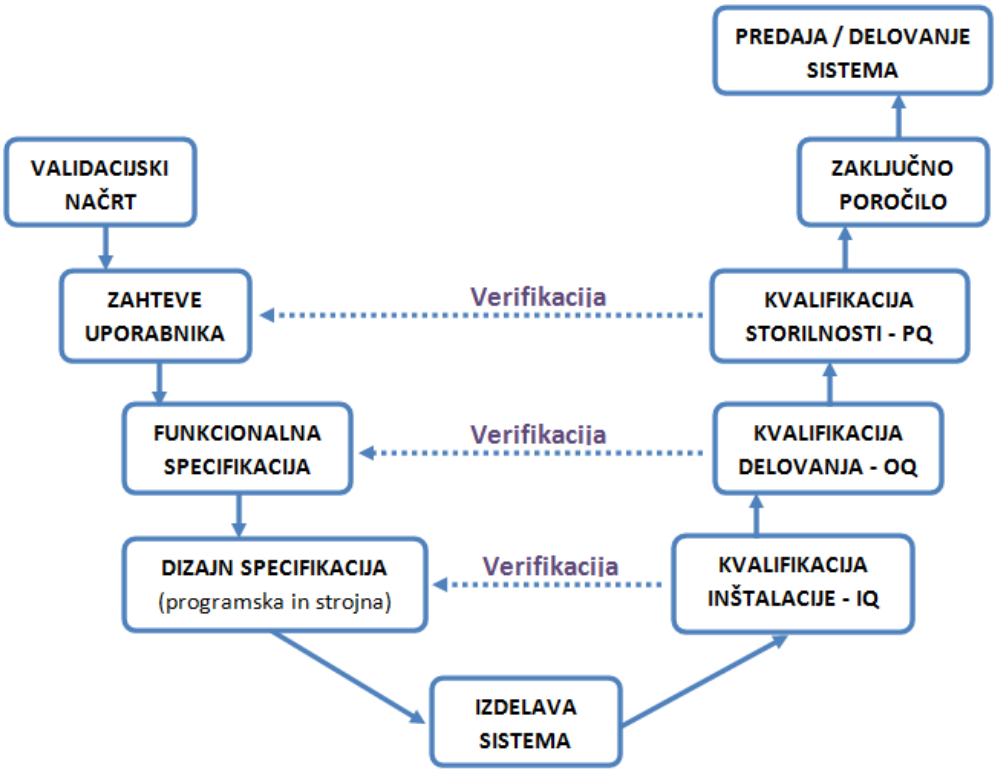
3.3 Izvedba validacije

Validacije se izvajajo v skladu z izhodišči GMP ter smernicami sektorja za upravljanje kakovosti. Validacijski cikel se prične z zapisom zahtev uporabnika, poteka preko priprave na izgradnjo, preko same izgradnje, izvedenih testiranj, predaje - prevzema sistema, vzdrževanja sistema v kvalificiranem stanju in njegove ukinitve.

Glavne aktivnosti validacije računalniških sistemov so:

1. planiranje,
2. specifikacije,
3. izgradnja sistema,
4. planiranje, testiranja in izvedba,
5. zaključki in poročila.

Posamezne aktivnosti si sledijo in so razvidne preko V modela. Preko sheme tega modela je tudi razvidno, katere aktivnosti si sledijo in kakšne so relacije med njimi. Shemo »V« modela lahko vidimo na sliki 16.



Slika 16: GAMPov »V« model verifikacije sistemov.

Potrebni dokumenti, ki v tem ciklu nastajajo so:

- VMP (validacijski master plan),
- zahteve uporabnika (ang. *User Requirements Specification* - URS),
- presoja dobavitelja,
- funkcionalna specifikacija (ang. *Functional Design Specification* - FDS),
- design specifikacija (ang. *Design Specification* - DS),
- kvalifikacija inštalacije (ang. *Instalation Qualification* - IQ),
- kvalifikacija delovanja (ang. *Operational Qualification* - OQ),
- kvalifikacija storilnosti (ang. *Performance qualification* - PQ),
- zaključna poročila sistema (delna in končna),
- zapisnik o izvedenem šolanju uporabnika,
- revalidacija – zapisnik revalidacije sistema,
- retrospektivna validacija – zapisnik retrospektivne validacije sistema,
- kontrola sprememb.

Zahteve uporabnika (User Requirements Specification - URS)

Zahteve uporabnika so dokument, v katerem so postavljeni želeni cilji, ki jih mora sistem dosegati. Dokument pripravi uporabnik, lahko pa ga pripravi tudi dobavitelj. Dokument mora biti pregledan in odobren s strani odgovornega uporabnika in odgovorne OQ osebe.

Validacijski master plan (VMP)

VMP mora definirati aktivnosti, procedure in odgovornosti za izvedbo posameznih aktivnosti v življenskem ciklu sistema. Prav tako je v VMPju definiran časovni plan posameznih aktivnosti. V VMPju se definira tudi način šifriranja dokumentacije, ki v življenjskem ciklu sistema nastaja. Odobren mora biti s strani vodstva podjetja, ki s svojim strinjanjem potrdi vire, potrebne pri izgradnji sistema.

Presoja dobavitelja

V primeru, da predstavlja računalniški sistem pretežni del sistema in je razvoj sistema delo zunanjih dobaviteljev, je potrebno opraviti presojo dobavitelja. Dobavitelj mora uporabljati sistem kakovosti, ki zagotavlja kakovost vgrajeno v sistem ter vso potrebno dokumentacijo. Certifikat o sistemu kakovosti (npr. ISO) ni zadosten dokaz, da je računalniški sistem kvalificiran in ne zagotavlja dovolj visoke stopnje dokumentacije za regulatorne organe.

Funkcionalna specifikacija (Functional Design Specification - FDS)

Funkcionalna specifikacija je dokument, ki ga pripravi izvajalec in s katerim odgovori na zapisane želje uporabnika definirane v Zahtevah uporabnika. Podpisana in odobrena (s strani uporabnika in dobavitelja) funkcionalna specifikacija, je osnova za začetek »Design« specifikacije. Funkcionalna specifikacija je hkrati tudi vodilo za pripravo funkcionalnega OQ testiranja sistema.

Design specifikacija (Design Specification - DS)

Design specifikacija je dokument, pripravljen s strani dobavitelja, ki vsebuje natančen opis rešitve (strojne in programske opreme) in implementacijo sistema v obstoječe okolje. Prav tako so v dokumentu opredeljeni načini programiranja in makro ter mikro pogoji, potrebni za nemoteno delovanje v realnem obstoječem okolju. Design specifikacija je vodilo za pripravo IQ procedur – preverjanje pravilnosti dejansko vgrajenih komponent. DS mora biti pregledana in odobrena s strani odgovornih oseb v podjetju.

IQ (Instalation Qualification)

IQ procedura mora biti pripravljena in pregledana pred izvedbo in odobrena po sami izvedbi. Odobrena in pregledana je s strani oseb definiranih v VMPju. S pomočjo IQ-ja preverimo (obseg aktivnosti je odvisen od samega sistema - poslovni, procesni sistem):

- ustreznost vgrajene opreme,
- pravilnost vgradnje opreme,
- pravilnost prenosa signalov od periferne opreme do prikaza na nadzornem sistemu,
- potrebne pogoje za pravilno delovanje sistema,
- vhodno izhodne enote (potrebna je kalibracija merilnih instrumentov),
- testiranje analognih in digitalnih signalov (kalibracija merilnih zank),
- preverjanje potrebnih dokumentov za nemoteno delo in vzdrževanje sistema,
- testiranje ustreznosti vgrajene programske opreme:
    - sistemska programska oprema,
    - aplikativna programska oprema,
- testiranje varnosti in zaščite.

Na koncu izvedenih aktivnosti sledi zaključno poročilo testiranja sistema.

OQ - Operational Qualification

OQ procedura mora biti pripravljena in pregledana pred izvedbo ter odobrena po sami izvedbi. Odobrena in pregledana je s strani oseb definiranih v VMPju. Pred izvedbo testiranj morajo biti vsaj v obliki osnutka pripravljeni SOPi, ki so potrebni za nemoteno delo s sistemom.

V sklopu testiranj OQ računalniškega sistema se izvajajo naslednja testiranja:

- testiranje funkcionalnega delovanja sistemov,
- testiranje sistema na stresne situacije – delovanje v kritičnih situacijah,
- testiranje ustreznosti zapisov podatkov v arhiv (zgodovina),
- testiranje delovanja »Audit traila«,
- testiranje ustreznosti sistema kontrole dostopov in zaščit sistema,
- testiranje poročil elektronskih zapisov ter elektronskih podpisov.

Na koncu izvedenih testiranj sledi zaključno poročilo funkcionalnega testiranja sistema.

PQ - Performance Qualification

V sklopu PQ testiranj se predvidi način, po katerem se bo v realnem delovnem okolju opazovalo in ocenilo obnašanje sistema skozi določeno obdobje (dogovorno določeno po izkušnjah in v odvisnosti od kritičnosti sistema). PQ teste pripravita skupaj izvajalec in lastnik sistema. PQ procedura mora biti pripravljena in pregledana pred izvedbo in odobrena po sami izvedbi. Odobrena in pregledana je s strani oseb definiranih v VMP-ju.

Zaključno poročilo

Zaključno poročilo je del validacijskega cikla, ki ga pripravi odgovorni uporabnik skupaj z izvajalcem. V njem se navedejo vse izvedene aktivnosti, ki dokazujejo pravilnosti delovanja sistema. Navede se tudi poročilo o izvedenem izobraževanju uporabnikov in vzdrževalcev sistema. Zapišejo se pomanjkljivosti v sistemu in odstopi ter način in rok za odpravo le-teh. Zapisnik podpišeta predstavnik izvajalca in predstavnik naročnika ter osebe definirane v VMP-ju.

Revalidacija sistema

Sistem je potrebno občasno revalidirati. Pogostost revalidacije je odvisna od kritičnosti in kompleksnosti samega sistema. V dokumentu »Revalidacija sistema« se predvidi, kateri testi so potrebni v življenjski dobi sistema, da bomo z njimi lahko potrdili kakovost delovanja sistema skozi njegovo življenjsko dobo. Za izvajanje revalidacije je odgovoren lastnik sistema.

Retrospektivna validacija

Gre za skupek aktivnosti potrebnih za dokazano pravilnost delovanja na starem – obstoječem sistemu. Validacija bazira na osnovi pregleda življenjskega cikla sistema in dokumentacije nastale v tem obdobju. Dodatno je podkrepljena z funkcionalnim testiranjem sistema, ter postopki potrebnimi za nemoteno delovanje in vzdrževanje sistema

Kontrola sprememb

Kontrola sprememb sistema je predpisan način vodenja sprememb na validiranih računalniško vodenih sistemih. Za vodenje kontrole sprememb je po predaji oziroma prevzemu odgovoren lastnik sistema.

4. Testiranje komponent sistema SDMS

Proces validacije vključuje tudi testiranje delovanja sistema. Če je sistem obsežnejši, se testiranja izvajajo po posameznih modulih, ali po logično zaključenih delih sistema. Vedeti moramo, da je samo izvajanje testiranj, v smislu poganjanja aplikacij na testnih podatkih, le manjši del opravil v procesu testiranja.

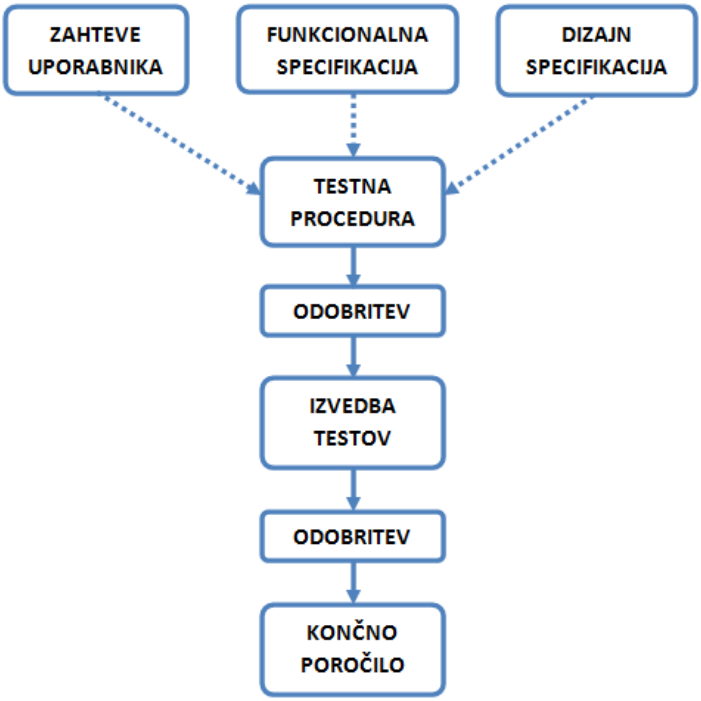
Da sem sploh lahko sodeloval pri testiranju dela SDMS sistema, sem moral spoznati proces validacije ter posamezne dokumente, ki pri tem nastajajo. Preučiti sem moral interne standarde podjetja Krka d.d., ki se nanašajo na področje zagotavljanja kakovosti ter upravljanja z računalniškimi sistemi. Preučil sem interne standarde, ki v osnovi izvirajo iz GAMPovih smernic [6]. GAMP (ang. *Good Automated Manufacturing Practice*) je namreč kratica, ki v angleščini pomeni »dobra praksa v avtomatizirani proizvodnji«. Smernice je izdala organizacija ISPE (ang. *International society for Pharmaceutical engineering*), ki je skupnost strokovnjakov s področja farmacije in si prizadeva za globalno širjenje znanj, tehnologije ter procesov v farmacevtski industriji. Pred tem sem natančno spoznal delovanje in lastnosti SDMS sistema preko uporabniških zahtev, ter funkcionalne specifikacije ponudnika sistema.

Celoten postopek svojega spoznavanja ter učenja sem moral sproti beležiti. Voditi sem moral nekakšen dnevnik učenja, kjer sem sproti zapisoval, kaj sem delal in koliko časa. Ves postopek mojega učenja je moral biti dokumentiran zaradi regulativ, ki v farmacevtski industriji obstajajo, ter zaradi zadostitve kvalitete dokumentacije za regulatorne organe.

4.1 Postopki v procesu testiranja

V Krki d.d. se vsi postopki testiranja začnejo s pripravo testnih procedur. Posamezno testno proceduro napiše skrbnik ali napredni uporabnik sistema. Testno proceduro avtor napiše na podlagi uporabniških zahtev, funkcionalne specifikacije ter dizajn specifikacije. Testne procedure nato pregleda ter odobri ali zavrne lastnik sistema. Ko so testi odobreni, se lahko preide na dejansko testiranje sistema. Teste lahko izvaja skrbnik sistema ali napreden uporabnik. V vlogi slednjega sem sistem testiral tudi jaz. Ko so testi izvedeni se jih pošlje v odobritev. Teste lahko odobri nekdo, ki je po položaju nad osebo, ki je izvajala teste.

Če so testi odobreni, se napiše poročilo, kjer se predstavi testno proceduro ter rezultate testa. V poročilu so prav tako zavedeni in podpisani vsi, ki so sodelovali v postopku testiranja. Izdelano poročilo se na koncu postopka pošlje sektorju za kakovost. Postopke v procesu testiranj vidimo na sliki 17.



Slika 17: Postopki v procesu testiranj.

4.2 Testne procedure

Testna procedura je dokument, v katerem so v logičnem zaporedju opisana testna navodila. Posamezen korak v testni proceduri je sestavljen iz testnih navodil, pričakovanega rezultata ter praznega polja kamor zapišemo rezultate testa.

V testni proceduri morajo biti zajeti tudi negativni testni primeri, s katerimi testiramo primere, ki ne smejo uspeti. Testna procedura mora biti skrbno izdelana, da zajame vse funkcionalnosti sistema ali dela sistema, ki se preverja. Primer enostavne testne procedure, za testiranje aplikacije »FilesToSDMSassistant«, prikazuje tabela 2.

Korak	Testna navodila	Pričakovan rezultat	Rezultat dosežen
1	Preveri nastavitve aplikacije FilesToSDMSassistant in jo zaženi.	Aplikacija se zažene.	
2	Na izvorno mesto pripravi podatke za analizno HPLC napravo in mapo s podatki za ostale neprave.	Podatki so v posameznih mapah.	
3	V aplikaciji FilesToSDMSassistant: - izberi način HPLC - izberi sekvenco v izvorni mapi/data, - izberi pripadajočo sdms datoteko - preveri rezultate prenosa v ponorni mapi	 Izbran način HPLC. Sekvenca v data mapi izbrana. Prenos datotek izveden. Datoteka izbrana. Premik izveden v podmape.	
4	Simuliraj postopek restora iz SDMS-ja tako, da ročno kopiraj vsebino iz ponora v izvor.	Vsebina je pripravljena za ponovitev postopka.	
5	Ponovi postopek prenosa za HPLC način za isto sekvenco. Aplikacija ugotovi, obstoj mape z enakim imenom: - Preglej vsebino ponora, - izvedi postopek, - preglej vsebino poora pred izbiro SDMS datoteke -preglej vsebino ponora po končanem postopku	 Prepis vsebine izvede tudi preimenovanje končnice SDMS datoteke. Po končanem postopku se v ponoru nahaja prenesena vsebina, stara sdms datoteka s preimenovano končnico in nova sdms daoteka.	
6	Simuliraj postopek restora iz SDMS-ja tako, da ročno PREMAKNI vsebino iz ponora v izvor.	Vsebina je pripravljena za ponovitev postopka.	
7	Ponovi postopek prenosa za HPLC način za isto sekvenco: - preglej vsebino ponora, - izvedi postopek, - preglej vsebino ponora pred izbiro SDMS datoteke -preglej vsebino ponora po končanem postopku	 Prenos vsebine izvede tudi preimenovanje končnice SDMS datoteke. Po končanem postopku se v ponoru nahaja prenesena vsebina, stare sdms datoteke s preimenovano končnico in nova sdms daoteka.	
8	Izvedi postopek prenosa za »Ostale naprave«	Prenos je uspešen, datoteka sdms se prenese v izbrani direktori v ponoru.	
9	Na oddaljenem računalniku z dodeljenim pristopom zajemi mapo z vsebino. Uporabi funkcijo »Oddaljen računalnik«	Mapa z vsebino se zajame.	
10	Ponovi predhodni .korak	Stara SDMS datoteka se preimenuje in prepiše z novo.	
11	Ponovi prenos za HPLC tako, da se izvor in ponor nahajata na omrežnem mestu.	Postopek se uspešno zaključi.	

Tabela 2: Primer enostavne testne procedure.

4.3 Nedokumentirano testiranje

Nedokumentirano testiranje se izvaja pred uradnim dokumentiranim testiranjem. Tovrstnega testiranja se poslužujemo takrat, ko od ponudnika dobimo prve verzije sistema ali aplikacije. Ko pri takem testiranju odkrijemo napako pri delovanju, ponudnika takega sistema informiramo o napaki, ki jo mora odpraviti. S tem postopkom pravilnost delovanja programa pripeljemo do te mere, da lahko izvedemo dokumentirano testiranje, ki je podprto z vso potrebno dokumentacijo. Z nedokumentiranim načinom testiranja skrajšamo čas, od dobave sistema, do uspešnega dokumentiranega testiranja. Čeprav nam bi nedokumentirano testiranje prineslo pozitivne rezultate, in bi bili prepričani v pravilost delovanja programa, je nujno izvesti še uradno dokumentirano testiranje. Nedokumentirano testiranje se od dokumentiranega loči po tem (poleg tega, da ga uradno ne dokumentiramo), da za testiranje ni nobenih testnih procedur. Oseba, ki izvaja nedokumentirano testiranje, si mora sama glede na URS ter na funkcionalno specifikacijo, izdelati testne scenarije. Z njihovo pomočjo lažje ter uspešneje izvaja testiranje. Testni scenariji iz nedokumentiranega testiranja služijo kot izhodišče za uradne testne procedure, po katerih se izvaja dokumentirano testiranje.

4.3.1 Zgled izvedbe nedokumentiranega testiranja

Pri svojem delu sem se srečal z nedokumentiranim testiranjem, ko sem testiral aplikacijo »FilesToSDMSassistant«. To aplikacijo je izdelal zunanji izvajalec, služi pa poenostavljanju prenosa podatkov. Ko oseba v laboratoriju zaključi s svojim delom in hoče podatke poslati v SDSM sistem, uporabi aplikacijo FilesToSDMSassistant, ki mu olajša delo. Uporabnik izbere podatke, ki jih želi shraniti v SDMS sistem, aplikacija »FilesToSDMSassistant« pa podatke premakne na lokacijo, iz katere zajemajo agenti za zajem podatkov.

Aplikacija ima tri načine delovanja:

- HPLC način je namenjen premikanju podatkov kromatografije,
- »Ostale naprave« je način delovanja, ki še najbolj spominja na »izreži in prilepi« postopek. Razlikuje se le v tem, da po končanem premikanju izberemo še ».sdms« datoteko, ki se premakne na najvišji nivo v ponorni mapi,

- »Oddaljen računalnik« je način delovanja aplikacije, pri katerem premikamo samo izbrano ».sdms« datoteko.

Najbolj kompleksen je prenos kromatografskih podatkov. Kromatograf rezultate analize shranjuje v tri mape:

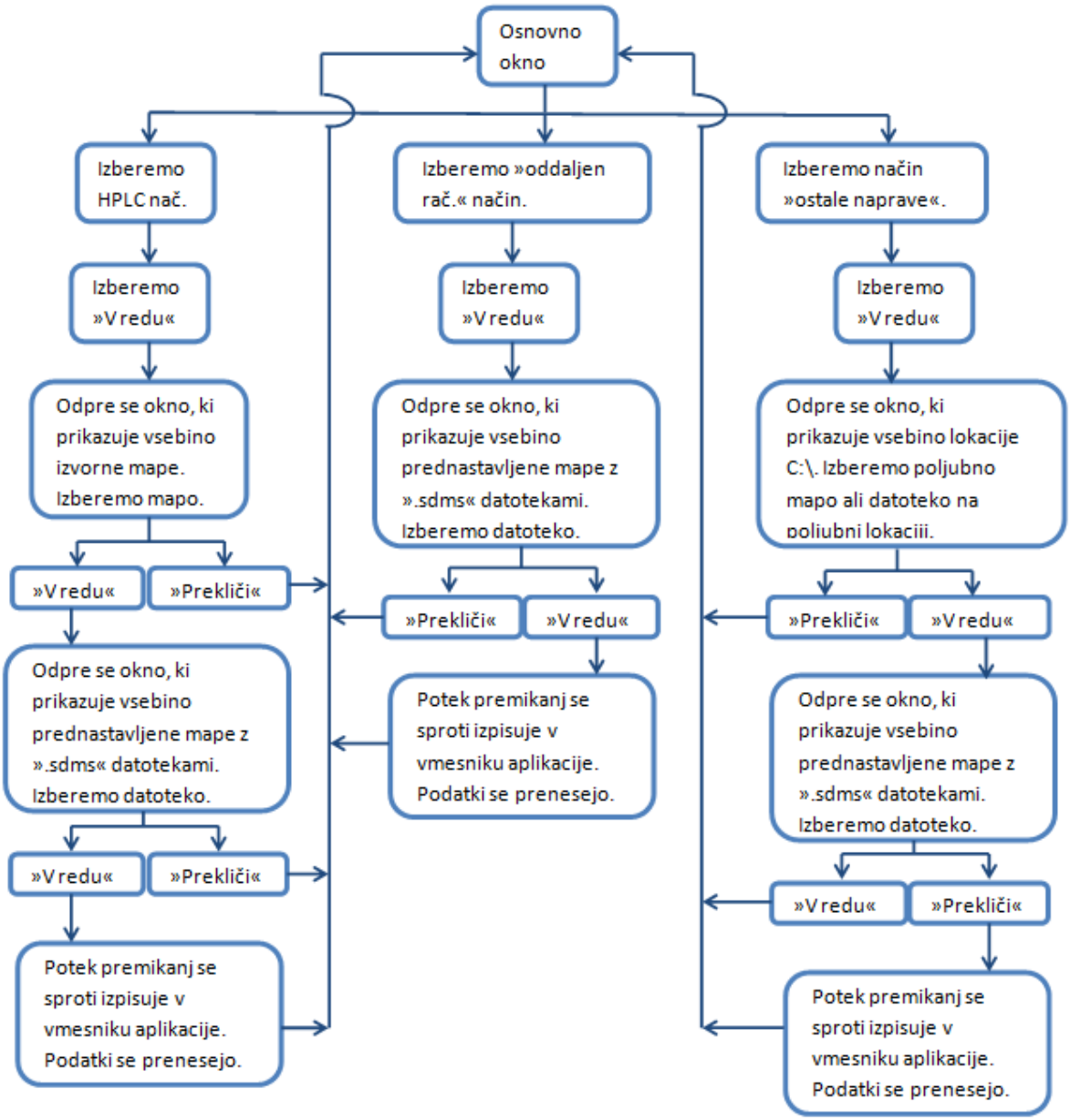
- »data« mapa vsebuje datoteke z rezultati kromatografskega postopka,
- »method« mapa vsebuje datoteke, ki opisujejo metode, ki so bile v postopku kromatografije uporabljene,
- »sequence« mapa vsebuje datoteke, ki opisujejo sekvence postopka kromatografije.

Če hočemo shraniti in na ta način objaviti podatke enega kromatografskega postopka v sistemu SDMS, moramo podatke shraniti na lokacijo dostopno SDMS agentom. Podatke posameznega kromatografskega postopka predstavljajo določene datoteke, iz vseh treh map. Zato je treba prenesti vse datoteke, ki pripadajo izbranemu kromatografskemu postopku. Uporabniku je pri uporabi aplikacije »FilesToSDMSassistant«, potrebno označiti le eno mapo v mapi »data« in aplikacija sama na določeno lokacijo premakne vse potrebne mape ter datoteke, ki pripadajo izbrani mapi. Glede na imena datotek in map, aplikacija prepozna, katere mape ter datoteke pripadajo določenemu kromatografskemu postopku.

Nastavitve za vsakega od treh načinov delovanja so shranjene v datoteki »Setting.xml«. V tej datoteki so definirane izvorne in ponorne mape, mape kjer se nahajajo ».sdms« datoteke ter sporočilo, ki se izpiše ob napaki. Primer take datoteke vidimo na sliki 18.

Slika 18: Vsebina datoteke z nastavitvami.

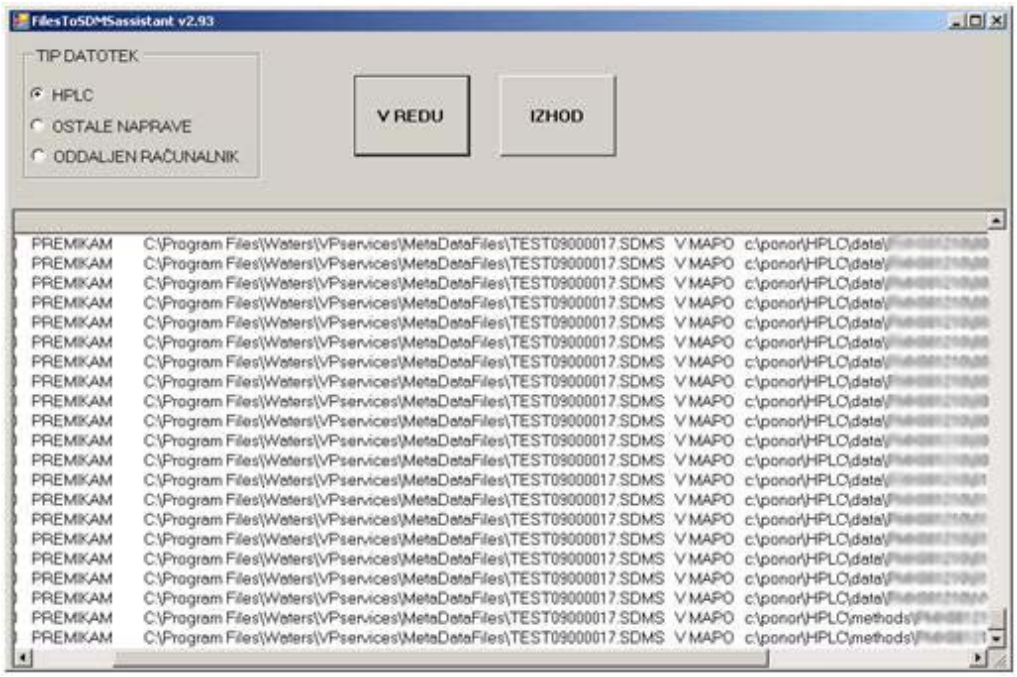
Preden sem začel testirati aplikacijo »FilesToSDMSassistant«, sem pregledal uporabniške zahteve zanjo. Glede na to, da sem izvajal nedokumentirano testiranje, sem moral napisati scenarij testiranja. Naredil sem si graf vseh možnih poti pri izvajanju aplikacije (slika 19).



Slika 19: Možne poti pri izvajanju programa »FilesToSDMSassistant«.

Testiranja sem razdelil na tri dele, za vsak način delovanja aplikacije posebej. Ker se morajo podatki pri vsaki izmed izbrahih možnosti, prenesti po določenih pravilih, sem temu posvetil še posebno pozornost. Med testiranjem sem odkril veliko nepravilnosti v delovanju. Sproti sem izvajalcu poročal o napakah, ta pa mi je pošiljal popravljene nove verzije aplikacije. Potrebnih je bilo veliko ponovitev testiranj.

Testiranja sem izvajal toliko časa, da sem ocenil, da aplikacija deluje v skladu z zahtevami. Ko je aplikacija delovala, kot je bilo predvideno v uporabniških zahtevah, sem pod nadzorom skrbnika sistema izvedel še dokumentirano testiranje. To sem lahko izvedel šele po tem, ko sva s skrbnikom sistema napisale testne procedure. Na sliki 20 imamo prikazan vmesnik aplikacije »FilesToSDMSassistant« po prenosu podatkov.



Slika 20: Vmesnik aplikacije »FilesToSDMSassistant«.

4.4 Dokumentirano testiranje

Pod nadzorom skrbnika sistema, sem sodeloval pri dokumentiranem OQ testiranju. Testirali smo del sistema SDMS in sicer zajem podatkov z instrumentalnimi agenti. Naj na tem mestu še enkrat omenim, da je pri testiranju računalniških sistemov 90 odstotkov časa namenjenega pripravi. Preučiti je treba uporabniške zahteve, funkcionalno specifikacijo ter dizajn specifikacijo. Na podlagi teh dokumentov je potrebno napisati testne procedure, ki v celoti preverijo funkcionalnosti sistema. Po tem sledi še proces odobritve testnih procedur, ter testnih rezultatov. Samo testiranje je bolj rutinsko delo, vendar pa še vedno zahteva osebo, ki pozna vse ostale postopke v procesu testiranja.

Testiral sem več kot 70 agentov, od katerih je vsak zajemal podatke iz različne naprave. Posamezne od teh naprav so proizvajale strukturno različne podatke, zato je bilo pomembno tudi to, da sem testiranja izvajal nad pravimi podatki. Na lokacije od koder so agenti zajemali podatke, sem ročno shranil realne podatke, ki jih je proizvedla določena naprava v drugem časovnem obdobju. Tako sem moral za vsako napravo zbrati realne testne podatke. Ko sem podatke shranil na lokacijo, iz katere jih pobirajo agenti, sem agenta pognal. Vsak testni korak sem moral sproti zabeležiti v dokumentaciji. Dokumente, ki so nastajali v procesu testiranj sem sproti objavljal na internem portalu. Za vsakega agenta sem moral sestaviti svoj dokument, v katerem je jasno prikazano kateri testni korak sem izvajal, ter ga podpreti s slikami zaslona. Slike so morale dokazati nedvoumnost izvedbe testnega koraka. Primer takega dokumenta je prikazan v prilogi tega dela. Najbolj pomembni sliki v tem dokumentu, sta tisti dve, ki prikazujeta vsebino map pred in po prenosu. Ti dve sliki vidimo v prilogi, na strani pet zgoraj ter na strani devet. Ostale slike prikazujejo preostale testne korake, ki so nas pripeljali do uspešnega prenosa, katerega predstavljajo prej omenjeni sliki. Testne procedure, po katerih sem testiral, so prikazane v tabeli 3.

Korak	Testna navodila	Pričakovan rezultat	Rezultat dosežen
1	Na računalniku preveri ime in ga vpiši. Ime preveriš tako da: *Na operacijskem sistemu klikneš Start->Nastavitve->Nadzorna plošča->System izbereš zavihek ime računalnika. *Na portalu podjetja poiščeš aplikacijo WhoAmI in jo zaženeš. Aplikacija se nahaja na: http://krkanet/Portal/Accessories/Applications/WhoAmI/	Ime računalnika s pomočjo opcije 1: _____ Ime računalnika s pomočjo opcije 2: _____	
2	Na računalniku preveri ali obstaja mapa na disku c:\ z imenom računalnika in podmapa SDMS, ki vsebuje mapi »save« in »error«. Mapi save ali error sta lahko vsebujejo elemete iz časa testiranj.	Obstajajo mape: C:\ »ime računalnika«\SDMS C:\ »ime računalnika«\SDMS\save C:\ »ime računalnika«\SDMS\error	
3	V aplikaciji Vision Publisher odpri dokument za kemijski, analizni ali biokemijski dnevnik. Dokument je lahko že izpolnjen oz. kreiraj novega in izpolni vsebino tako, da vsebuje vse metapodatke. Preveriš jih, da v odprtem dokumentu izbereš Properties->Document v oknu izbereš zavihek Metadata.	Metapodatki so definirani.	

Korak	Testna navodila	Pričakovan rezultat	Rezultat dosežen
4	V dokumentu odpreš Sekcijo 1 oz. Glavo dokumenta. V njej se nahaja gumb »Prenos v SDMS«. S klikom na gumb se kreira datoteka z imenom dokumenta in končnico *.sdms. Odpre se okno aplikacije »FilesToSDMSassistant« v katerem izberemo način za zajem metapodatkov. Običajno izberemo Ostale naprave oz. HPLC za analizne naprave. Funkcije za oddaljen računalnik priporočam le naprednejšim uporabnikom. Nadaljujemo z »V redu«.	Datoteka se *sdms kreira. Odpre se okno aplikacije.	
5	V iskalniku izbereš mapo,ki pripada testirani napravi. Z raziskovalcem preveri vsebino. Nadaljuješ »V redu«	Mapa je definirana. Podatki so veljavni.	
6	Izberemo datoteko katere ime je enako odprtemu dokumentu v laboratorijskem dnevniku Nadaljuješ z »Odpri«.	Datoteka je izbrana.	
7	Datoteke se prenesejo na mesto definiran v koraku 2.	Mapa se skopira in ustvari pogoj za zagon postopka zajema. Kreira se *.zip datoteka. z vsebino mape Agent zajame datoteko *.sdms in izlušči metapodatke. Podatke in metapodatke zajame Data Adapter ter shrani v podatkovno bazo	
8	Zaženi aplikacijo SDMS Vision. Izvedi prijavo, ki omogoča vpogled v projekte »FileDataRD« in »PrintDataRD«.	Aplikacija se zažene. Prijava uspe. Projekti so vidni.	
9	Izberi projekt »FileDataRD«, pogled »Default View«, sortiraj po ID naraščajoče. Nekajkrat osveži zaslon z »Refresh« dokler se ne pojavi novo zajeta datoteka. Za iskanje pravega dokumenta lahko uporabiš funkcije filtra ali iskanja.	Pojavi se *.zip datoteka z imenom mape.	
10	Preglej zajete meta podatke in ugotovi ali se skladajo z definiranimi metapodatki v *.SDMS Datoteki.	Metapodatki so zajeti.	
11	Označi dokument in klikni na »Preview«.	Prikaže se vsebina *.zip datoteke, enaka vsebini pred zajemom. Prazne mape niso zajete. Vsebina je identična.	

Tabela 3: Testne procedure za testiranje zajema podatkov.

4.4.1 Testiranje zajemanja podatkov iz ostalih naprav

Pod pojmom »ostale naprave« mislim na laboratorijske naprave, ki niso HPLC naprave. Testiral sem jih po enaki testni proceduri kot HPLC naprave (glej tabelo 3). Primer dokumentiranega testa po proceduri iz tabele 3, je priložen v prilogah.

Zajemanja sem prožil iz aplikacije »Vision Publisher«. V aplikaciji sem odprl že izpolnjen laboratorijski dnevnik, kjer sem s klikom na gumb »Prenos v SDMS« sprožil prenos. Po kliku na gumb se je ustvarila datoteka s končnico ».sdms«. V tej datoteki so bili shranjeni metapodatki iz laboratorijskega dnevnika. Po tem, ko se je ustvarila ».sdms« datoteka, se je pognala aplikacija »FilesToSDMSassistant«. V tej aplikaciji sem izbral podatke, ki so pripadale napravi, katere agenta sem testiral, ter novoustvarjeno ».sdms« datoteko. Aplikacija »FilesToSDMSassistant« je opravila premik podatkov na predvideno lokacijo. Premik se je izvršil tako, da so se najprej premakniki podatki testiranj, nazadnje pa se je prenesla ».sdms« datoteka. Tak vrstni red opravil je moral biti določen zato, ker je v nastavitvah agentov, ».sdms« datoteka nastavljena kot sprožitvena datoteka. Ko je agent našel ».sdms« datoteko, je začel z zajemanjem. Po določenem času so se podatki zajeli. V spletni aplikaciji VISION, se je pojavil nov zapis z imenom mape, katere podatke sem zajel (ime mape, ki jo v izberemo aplikaciji »FilesToSDMSassistant«). S klikom na zapis se nam odpre okno, ki prikazuje vsebino zapisa. Nujno je, da se podatki na izvoru in ponoru ujemajo.

Pri zajemanju iz ostalih naprav ni bilo primera, ko test ne bi bil uspešen. Deloma gre to pripisati enostavni strukturi podatkov, ki jih zajemamo. Večina zaslug za uspešno testiranje pa gre skrbniku sistema, ki je dobro nastavil agente.

4.4.2 Testiranje zajemanja podatkov iz HPLC naprav

Pri HPLC napravah, ki proizvajajo kompleksne podatke, je veliko faktorjev, ki vplivajo na uspešnost rezultatov testiranj. Sam uspeh testiranj je torej močno odvisen od drugih gradnikov sistema. V nekaterih primerih testiranj, so se podatki v SDMS sistem uspešno prenesli, težave pa so se pojavile kje drugje. Ena od težav je bila, da se vsebina map v spletni aplikaciji VISION ni prikazovala. To se je dogajalo za podatke, katere prikazujemo v obliki grafa. Ta napaka ni bila povezana z samimi agenti za zajem podatkov. Napaka je bila v nastavitvah strežnika na katerem teče aplikacija VISION. Ko sem tako napako odkril, sem skrbnika sistema opozoril nanjo. Skrbnik sistema je ukrepal po postopkih, ki so predvideni za take primere. Ta čas sem izvajal teste drugih agentov. Agente, ki so povzročali težave v aplikaciji VISION namreč nisem mogel preizkusiti, ker nisem mogel izvesti zadnjega testnega koraka (glej tabelo 3).

Med samim testiranjem sem odkril tudi nekaj težav z delovanjem agentov, ki so bile posledica napačnih nastavitev. Tudi o teh težavah sem poročal skrbniku sistema. Če sem odkril vzrok napake, sem nastavitev popravil sam (o tem sem obvestil skrbika sistema), drugače pa je to storil skrbnik. V večini primerov, se je določena napaka v nastavitvah pojavljala pri več agentih, zato njeno odpravljanje ni predstavljalo večjega problema.

4.5 Rezultat testiranj

Testiranja so bila uspešna, saj so bili uspešno izvedeni vsi predvideni testi. Tudi pri testih, kjer smo našli napake, smo napako popravili in izvedli novo testiranje. Uspešno je bilo tako nedokumentirano kot tudi dokumentirano testiranje. Nedokumentirano je bilo uspešno v smislu tega, da sem pravilnost delovanja pripeljal do točke, od katere dalje, se je lahko dokumentirano testiralo. Na ta način, sem prihranil nekaj časa in truda, v primerjavi s tem, da bi aplikacijo testirali dokumentirano takoj, ko nam bi jo izvajalec dostavil. Uspešno izvedeno OQ testiranje zajema podatkov s pomočjo agentov, je bilo zadnje v vrsti testiranj, preden je sistem prešel v uporabo.

Razlog za uspešna testiranja je bila dobra dokumentacija, vse od URSja naprej. Tudi pravilne testne procedure so pripomogle k temu, da je sistem kvalitetno stestiran, kar pomeni, da je možnost napak v prihodnosti čimanjša. K uspešnosti testiranj pa smo nenazadnje pripomogli tudi vsi, ki smo s svojim znanjem ter doslednostjo izvajali teste.

5 Zaključek

Opravljena testiranja so bila izvedena v sklopu validacije sistema SDMS. Izvajali smo OQ teste, ki so zadnji v vrsti aktivnosti validacije sistema, preden se sistem zažene v realnosti. Zelo pomembno je, da so testiranja uspešno izvedena. V nasprotnem primeru, ko se pojavijo napake, je te treba odpraviti. Odvisno od velikosti problema se napako dlje ali manj časa odpravlja, kar prinese dodatne stroške in podaljša izvedbo projekta.

Če odmislim nekaj primerov, ko smo pri testiranjih odkrili manjše napake, je bilo testiranje uspešno. Za vse opravljene teste smo uredili dokumentacijo, skrbnik sistema pa je sistem predal v izvajanje.

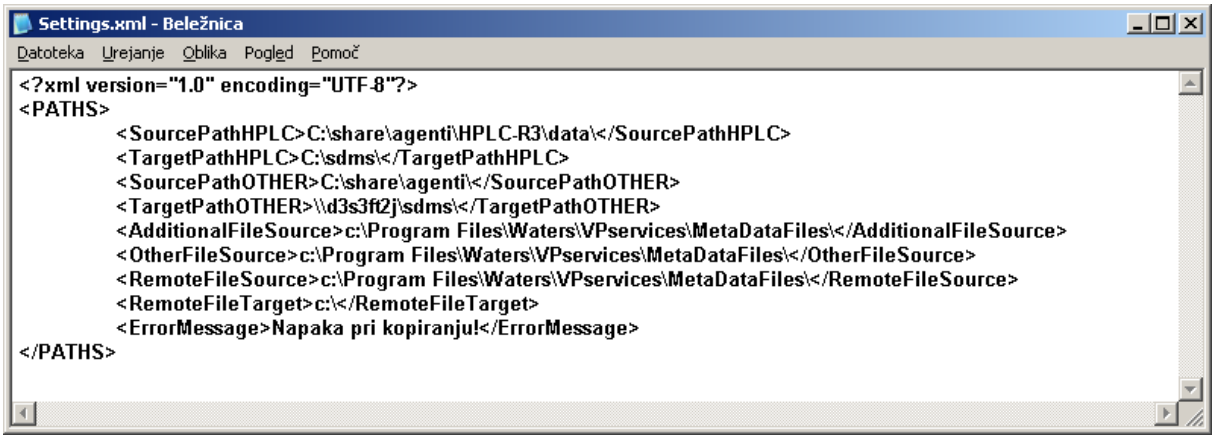
Pri procesu validacije nastane ogromno dokumentacije, govorimo o več deset fasciklih. Na tem področju, bi bilo dobro zasnovati tak sistem, ki bi omogočal lažjo izdelavo ter upravljanje dokumentacije.

Samo testiranje je proces, kjer ni bljižnic, zato je potrebno delati po postopkih in predpisih, ki zanj veljajo. Sam ne vidim velikih možnosti za optimizacijo testnega procesa, so pa kakšne malenkosti, ki bi se jih dalo izboljšati.

Priloge

Priložena je priloga preizkusa agenta »PKR_106632_automate«.

Nastavitve aplikacije »FilesToSDMSAsisstant«.

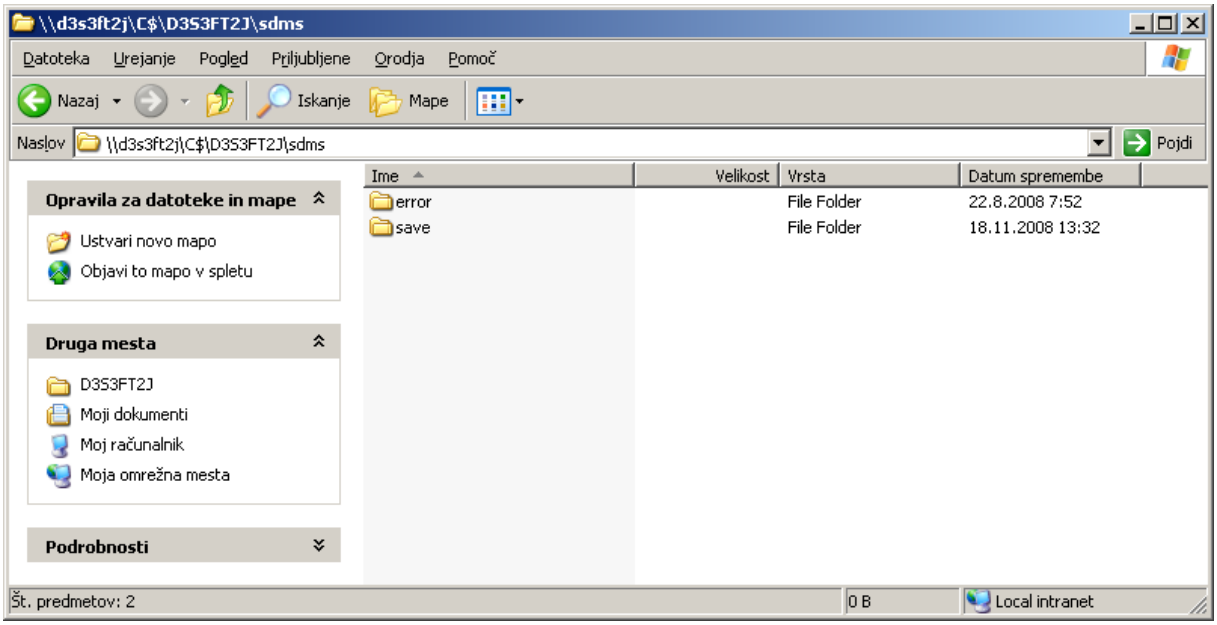


Korak1:

Ime računalnika: \\d3s3ft2j

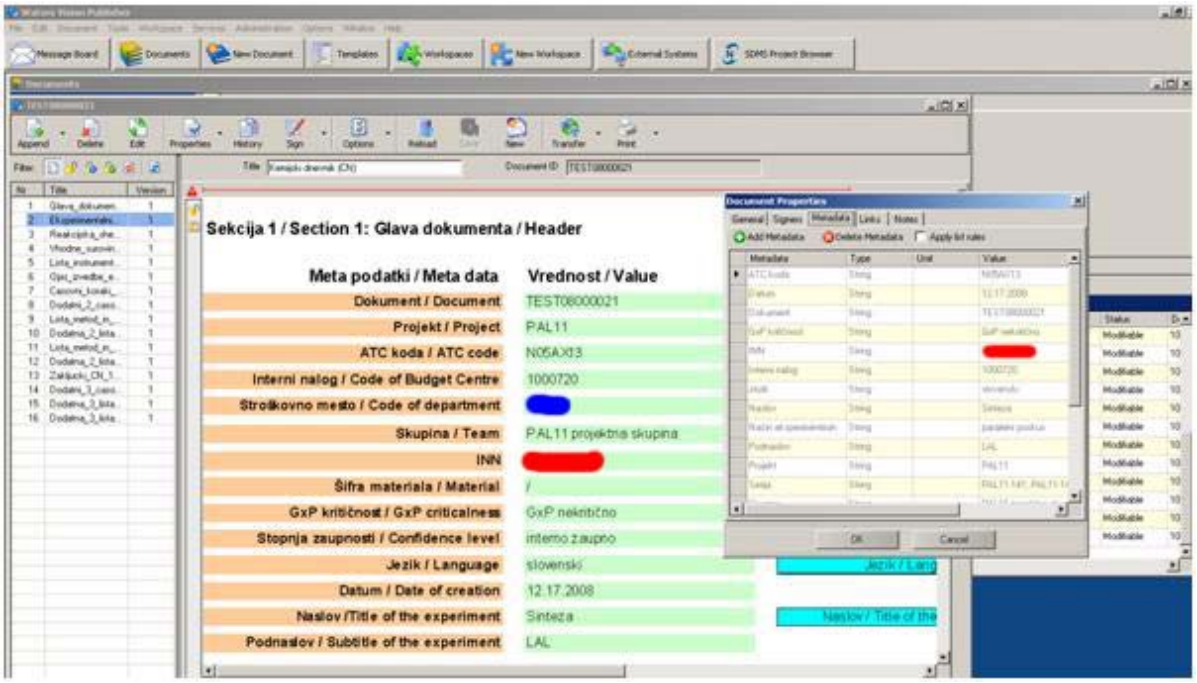
Korak 2:

Mapa SDMS obstaja

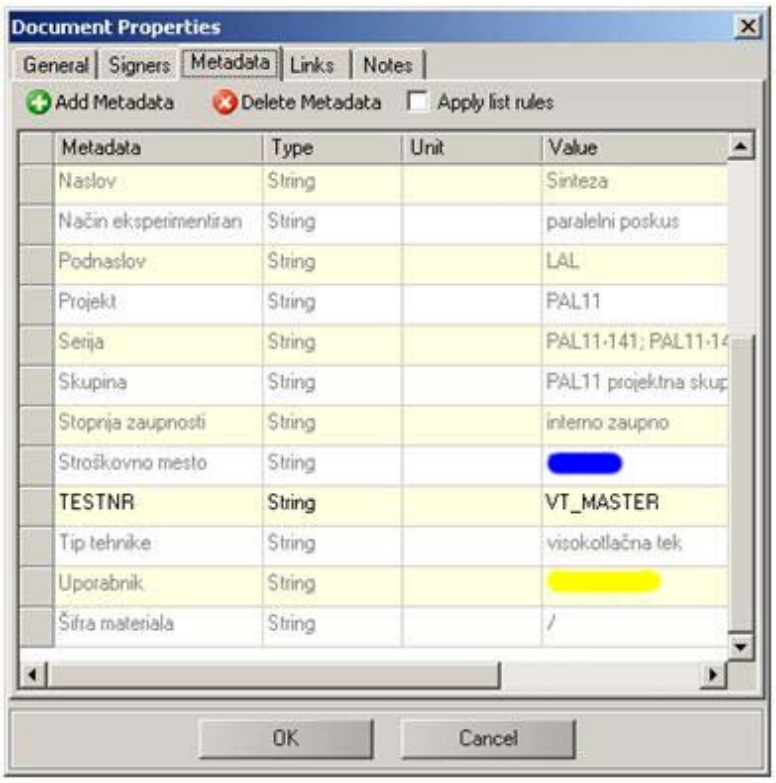


Korak 3:

Dokument je izpolnjen

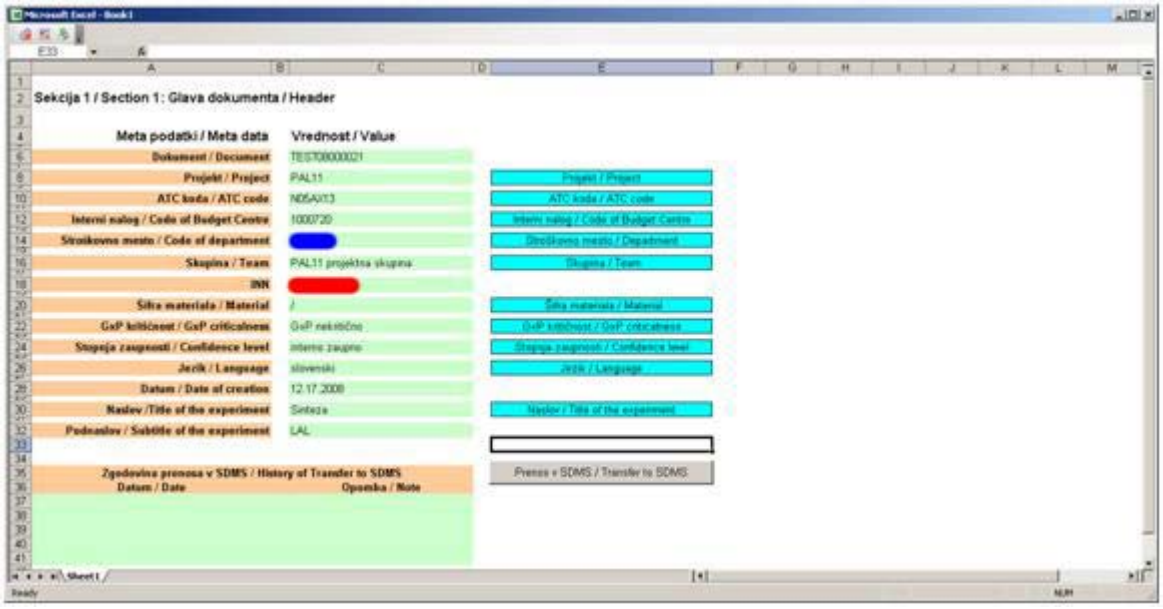


Definirani metapodatki v laboratorijskem dnevniku.

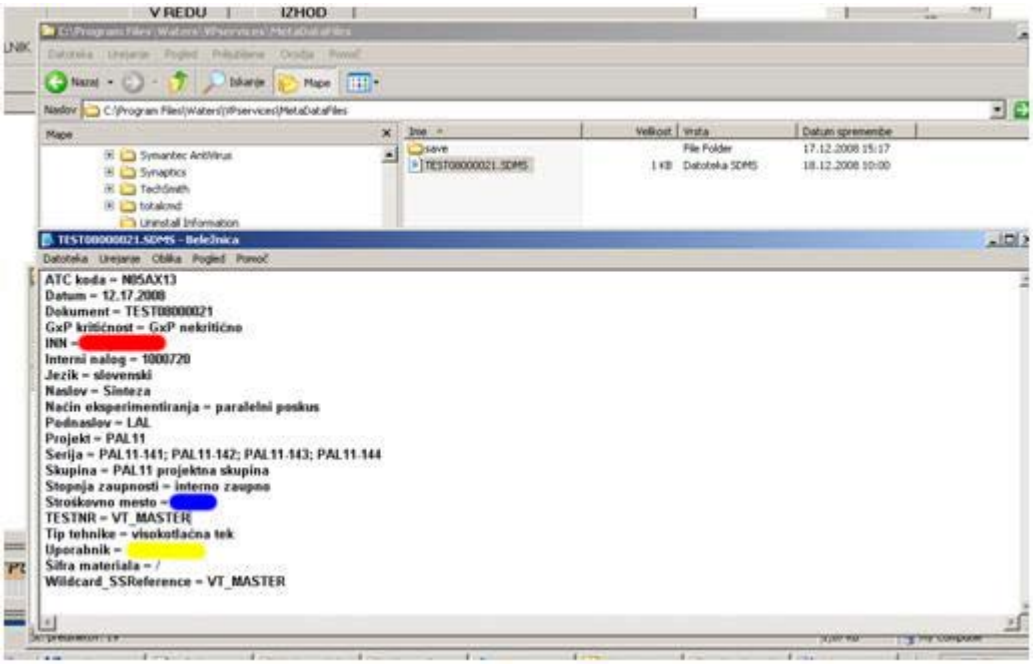


Korak 4:

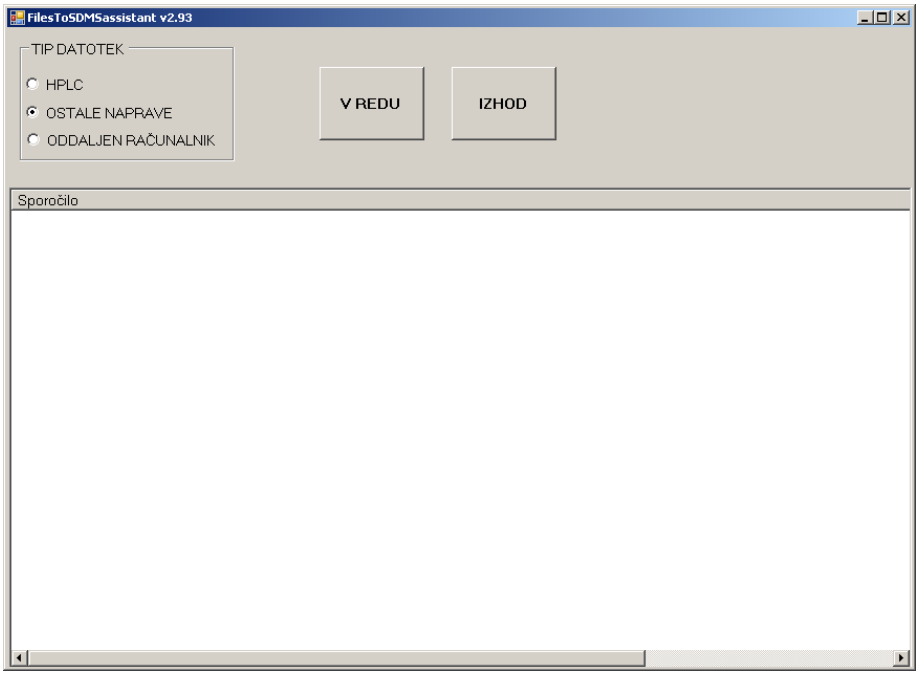
Sekcija je aktivna:



Po kliku se kreira »*.sdms« datoteka.

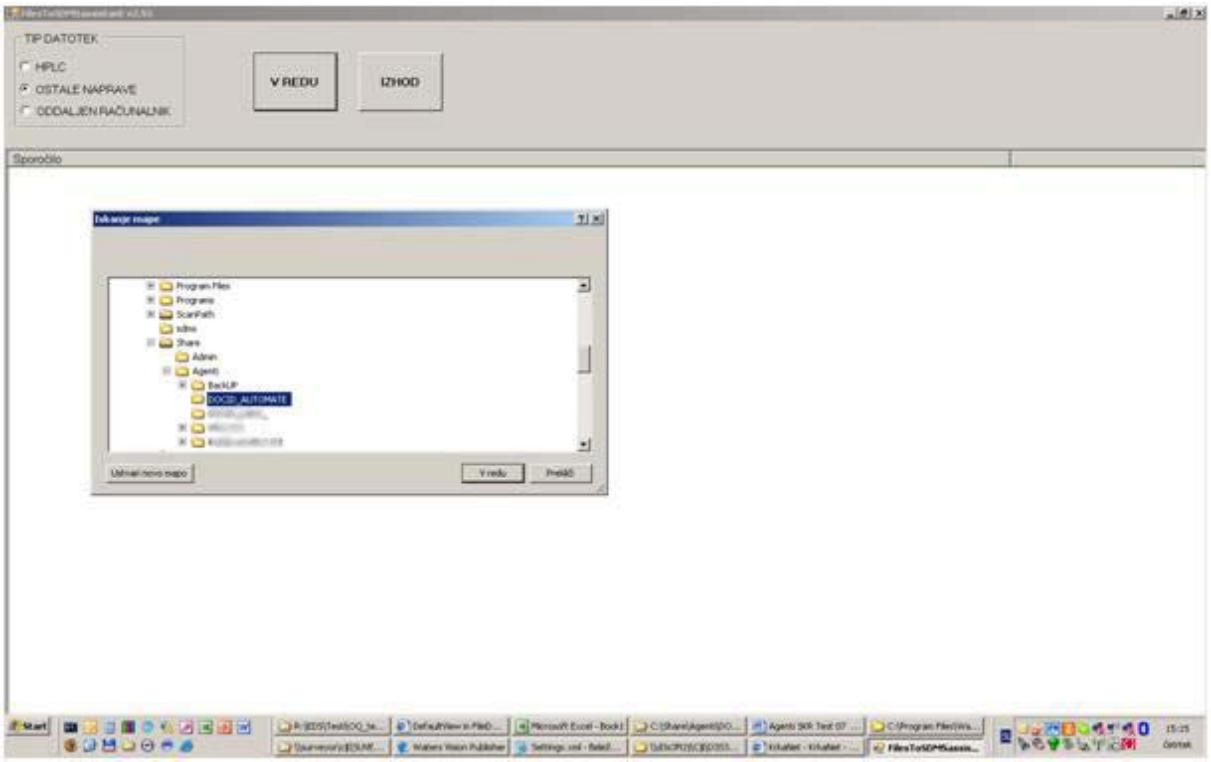


Aplikacija za kopiranje datotek/map se zažene. Izberemo »Ostale naprave«.

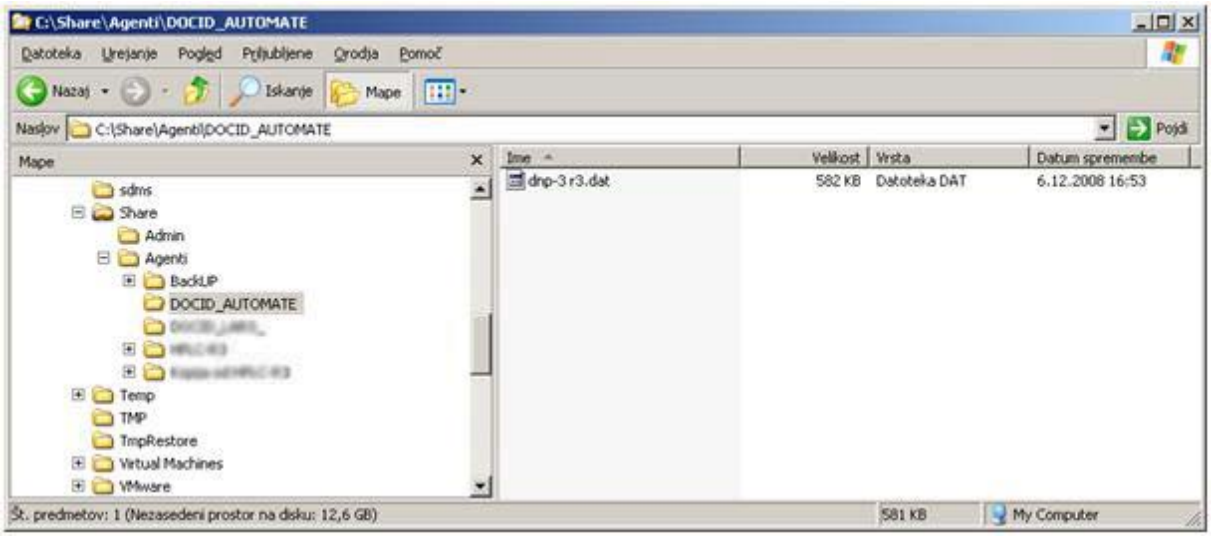


Korak 5:

Izberemo mapo s testnimi podatki za pripadajočega agenta.

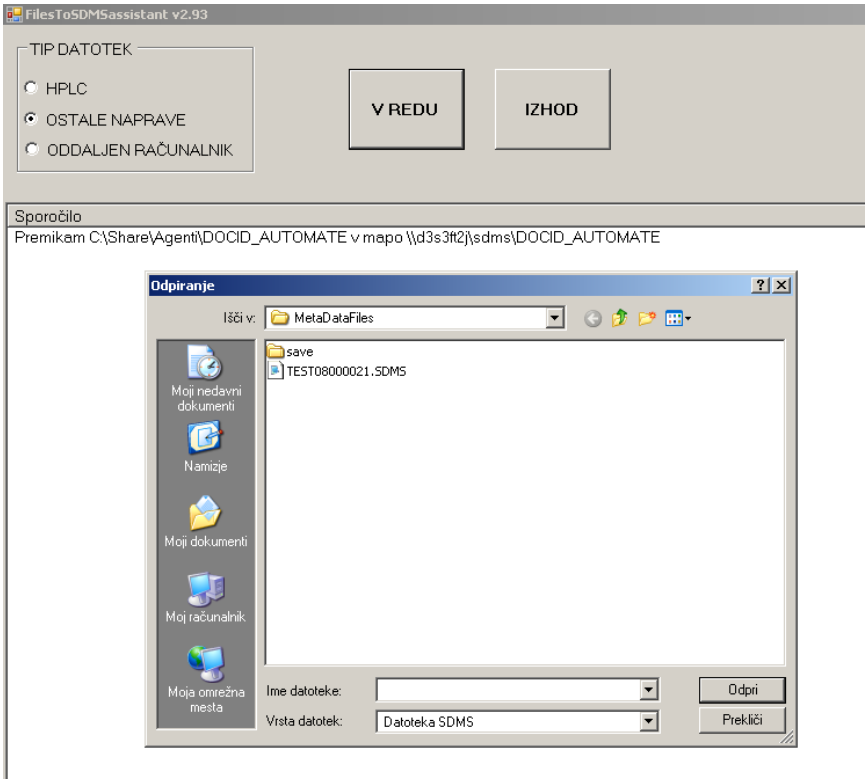


Vsebina mape.



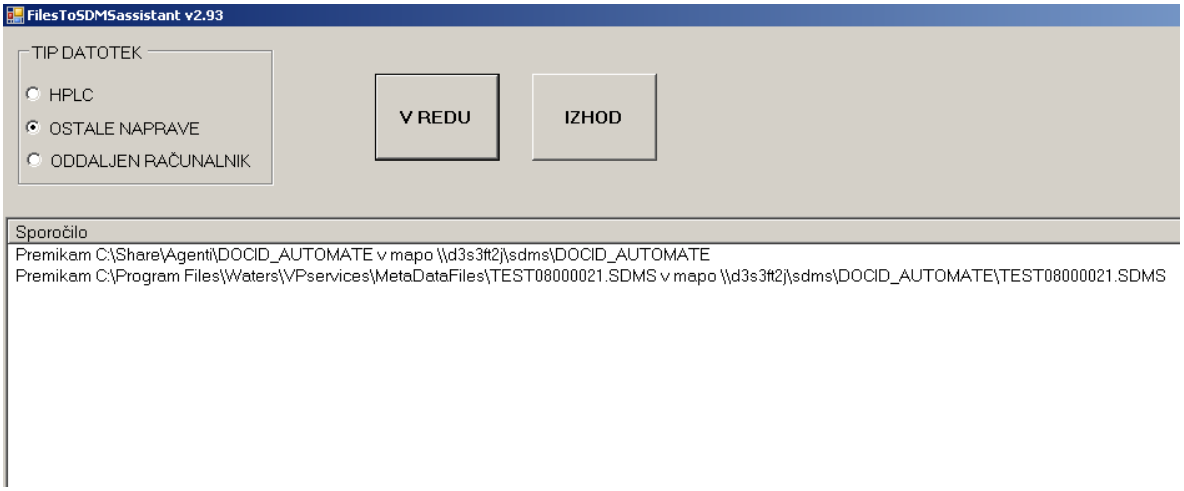
Korak 6:

Izberemo »*.sdms« datoteko, ki pripada odprtemu dokumentu v laboratorijskem dnevniku.



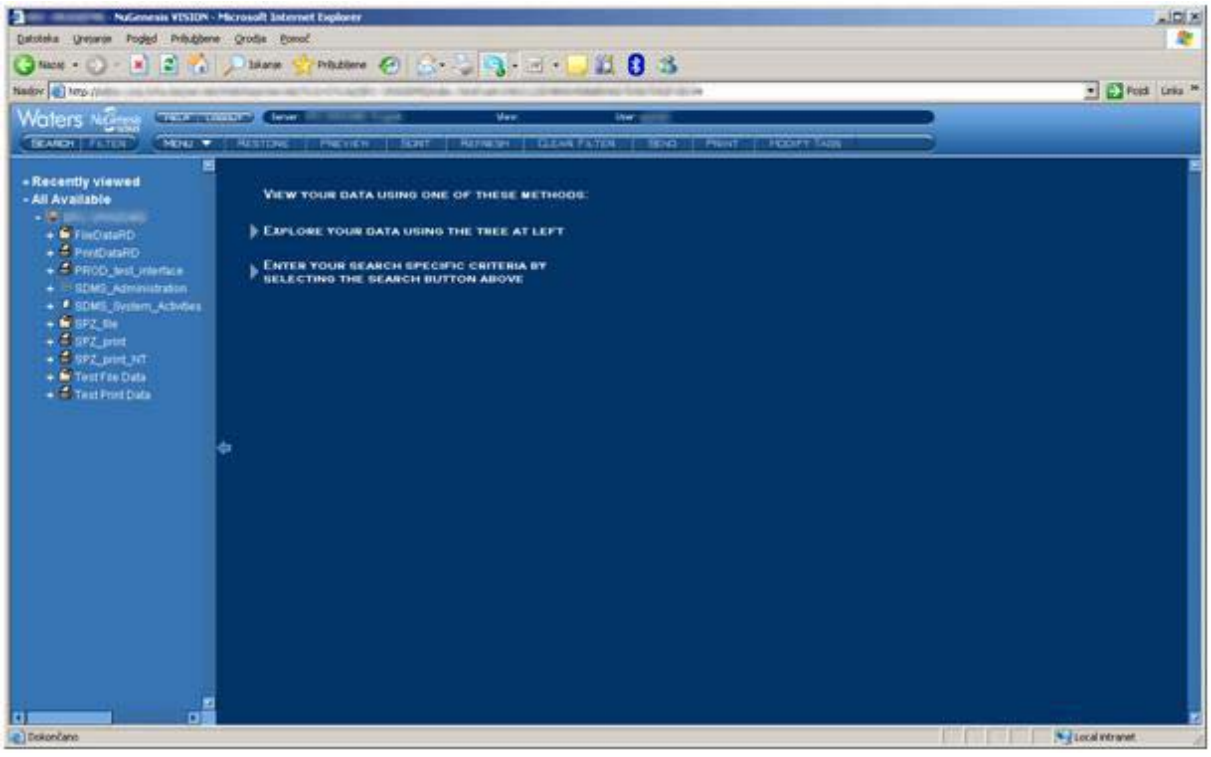
Korak 7:

Sporočilo aplikacije po končanem prenosu.



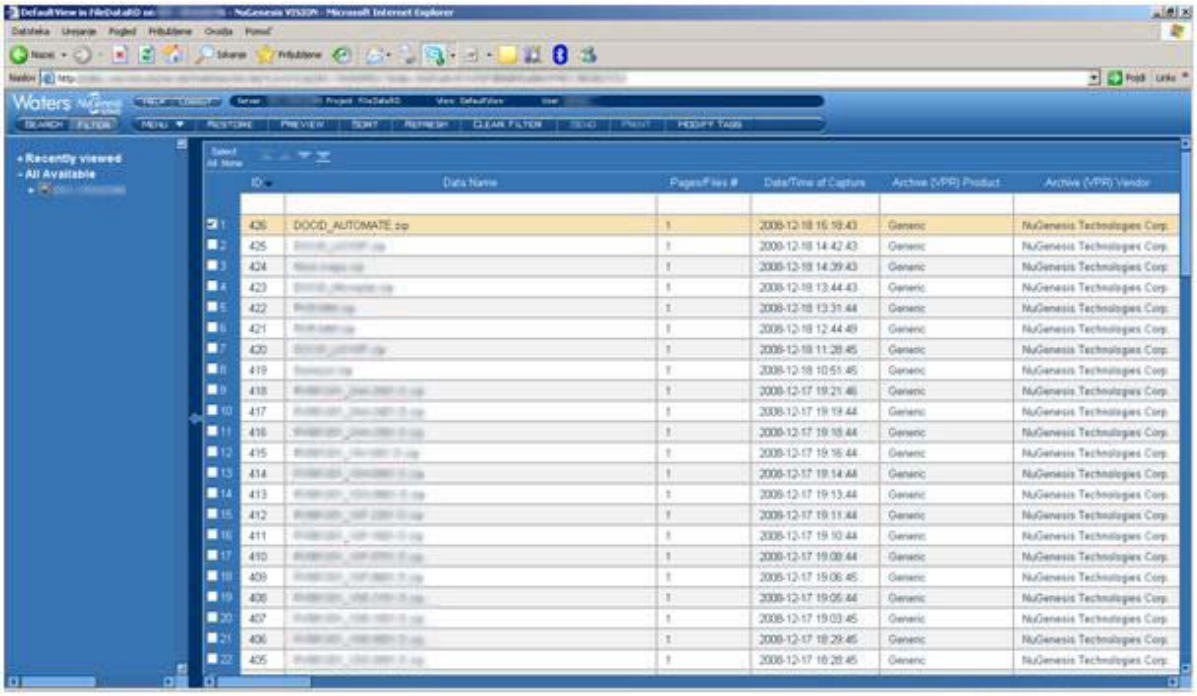
Korak 8:

SDMS Vision aplikacija.



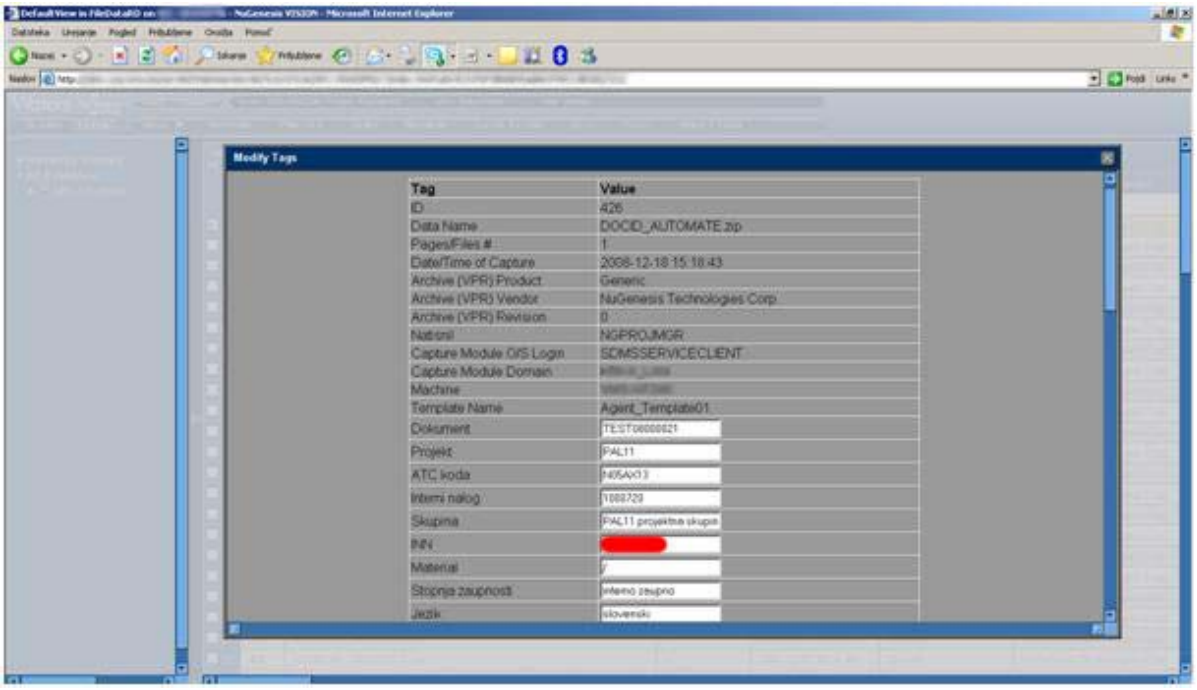
Korak 9:

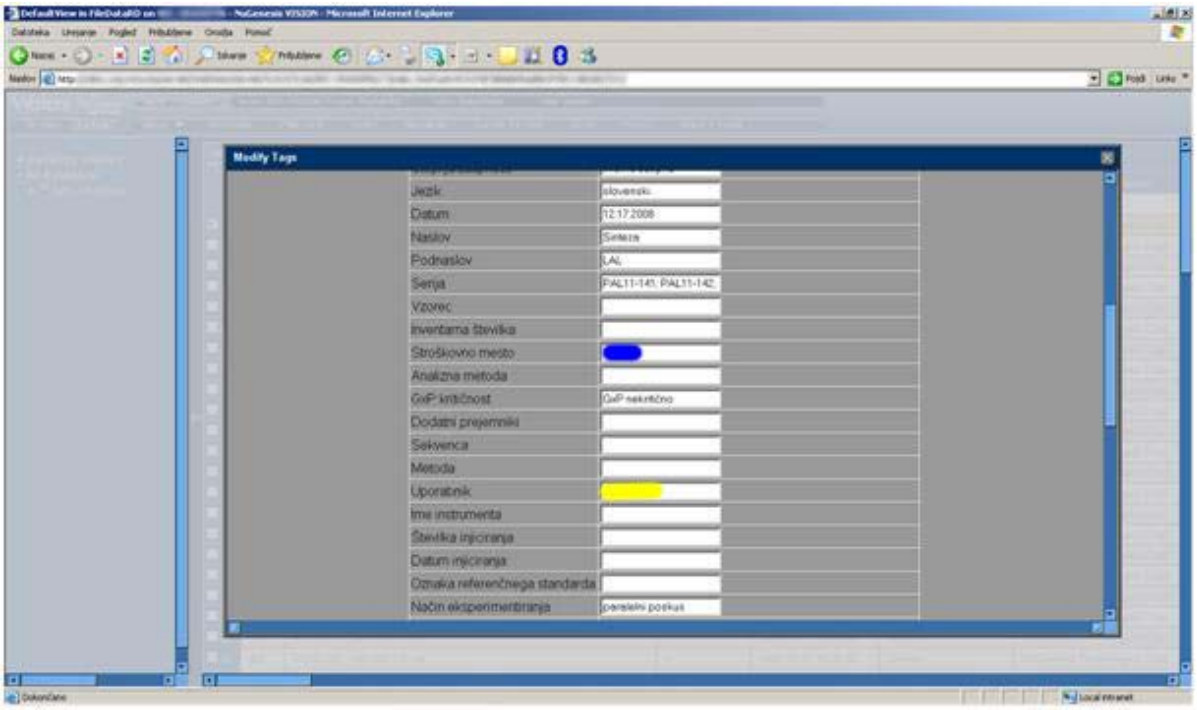
Podatki so zajeti.

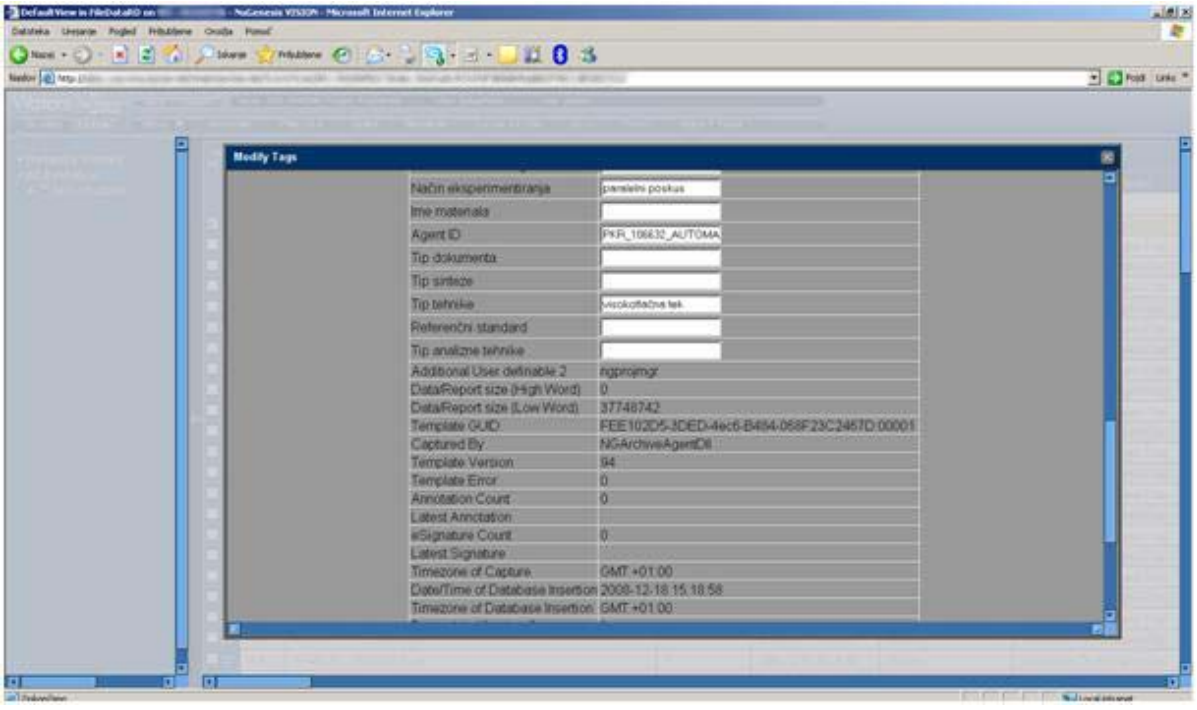


Korak 10:

Metapodatki.

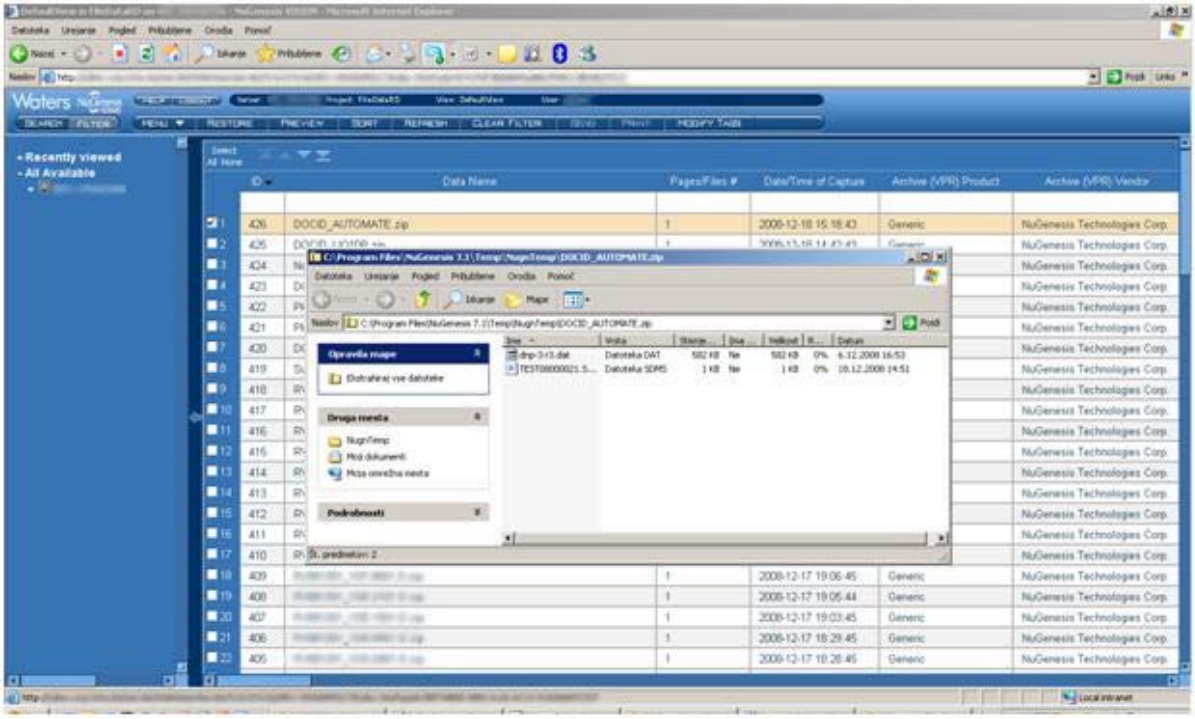



Modify Tags
Jezik
slovenski
Datum
12.17.2008
Naslov
Sinteza
Podnaslov
LAL
Serija
PAL11-141, PAL11-142
Vzorec
Inventarna številka
Stroškovno mesto
Analizna metoda
GxP kritičnost
GxP nekritično
Dodatni prejemniki
Sekvenca
Metoda
Uporabnik
Ime instrumenta
Številka injiciranja
Datum injiciranja
Oznaka referenčnega standarda
Način eksperimentiranja
paralelni poskus


Modify Tags
Način eksperimentiranja
paralelni poskus
Ime materiala
Agent ID
PKR_106632_AUTOMA
Tip dokumenta
Tip sinteze
Tip tehnike
visokotlačna tek.
Referenčni standard
Tip analizne tehnike
Additional User definable 2
ngprogr
DataReport size (High Word)
0
DataReport size (Low Word)
37748742
Template GUID
FEE102D5-3DED-4ec6-B484-068F23C2467D:00001
Captured By
NGArchiveAgentDll
Template Version
94
Template Error
0
Annotation Count
0
Latest Annotation
eSignature Count
0
Latest Signature
Timezone of Capture
GMT +01:00
DateTime of Database Insertion
2008-12-18 15:18:58
Timezone of Database Insertion
GMT +01:00

Korak 11:

Vsebina ustreza.



Viri

[1] **Victoria Lander.** Cycle time reduction in manufacturing using a scientific data management system. [Elektronski] April 2004. http://www.21cfrpart11.com/files/library/compliance/cycle_time_red_chemtoday04_05_2004.pdf.

[2] **21 CFR Part 11 Final rule.** Department of Health and Human Services; Food and Drug Administration. [Elektronski] 20 marec 1997. http://www.21cfrpart11.com/files/library/government/21cfrpart11_final_rule.pdf.

[3] **Mario Valle.** Scientific Data Management – an introduction. [Elektronski] 16 april 2008. http://personal.cscs.ch/~mvalle/sdm/scientific-data-management.html.

[4] **Waters Corporation.** Laboratory informatics business solutions. April 2008. 720002493EN LB-AP.

[5] **Jim Gray, David T. Liu, Maria Nieto-Santisteban, Alexander S. Szalay, David DeWitt idr..** Scientific Data Management in the Comming Decade. Redmond, ZDA: Microsoft Research, januar 2005.

[6] **ISPE, GAMP Guide for Validation of Automated Systems**, GAMP 5, Februar 2008.